METHOD AND SYSTEM FOR THE DISTRIBUTED CREATION OF A PROGRAM FOR A PROGRAMMABLE PORTABLE DATA CARRIER

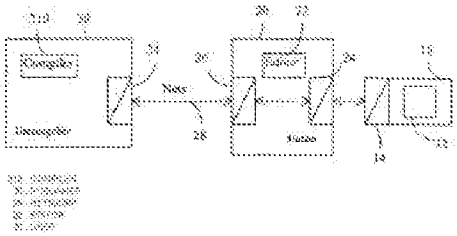
Publication number: WO02069118 (A2)
Publication date: 2002-09-06
Inventor(s): GOLLNER MICHAEL [DE]; CIESINGER DANIEL [DE] +
Applicant(s): GIESECKE & DEVRIENT GMBH [DE]; GOLLNER MICHAEL [DE]; CIESINGER DANIEL [DE] +
Classification:
- international: ***G06F1/00; G06F21/00; G06F9/44; G06F9/45; G06K17/00; G06K19/07;*** **G06F1/00; G06F21/00; G06F9/44; G06F9/45; G06K17/00; G06K19/07;** (IPC1-7): G06F1/00
- European: G06F21/00N1C; G06F21/00N1D1; G06F21/00N3P1; G06F21/00N9C; G06F21/00N9S; G06F21/00N9T; G06F9/45
Application number: WO2002EP01655 20020215
Priority number(s): DE20011008487 20010222

Also published as:
WO02069118 (A3)
US2004148502 (A1)
RU2289157 (C2)
JP2004528632 (T)
EP1393146 (A2)
more >>

Cited documents:
US6005942 (A)
EP0778522 (A2)
US5313635 (A)
EP0464526 (A2)
XP004304910 (A)

Abstract of **WO 02069118 (A2)**

The invention relates to a method for the distributed creation of a program for a programmable portable data carrier (10), for example, a chip card. To this end, program source text (Q) is created on a user computer (20), compiled and linked to executable program code (C) on a spatially separate compiler server (30), and the executable program code (C) is loaded into the data carrier (10) once again via the user computer (20). A secure end-to-end link is established for conducting an exchange of data between the data carrier (10) and the compiler server (30). To this end, the data carrier (10) is provided, in a pre-completion step, with software tools for final processing, which permit a transport code (U, Cssl, UCSM) provided in a transition format to be converted into executable program code (C). The transport code (U, Cssl, UCSM) is secured by encoding mechanisms.; The transmission of the executable program code (C), which is generated by the compiler server (30), ensues in the transition format (U, Cssl, UCSM).



Data supplied from the ***espacenet*** database — Worldwide

Europäisches Patentamt
European Patent Office
Office européen des brevets

Description of WO02069118	Print	Copy	Contact Us	Close

Result Page

Notice: This translation is produced by an automated process; it is intended only to make the technical content of the original document sufficiently clear in the target language. This service is not a replacement for professional translation services. The esp@cenet® Terms and Conditions of use are also applicable to the use of the translation tool and the results derived therefrom.

Method and system to the distributed creation of a programme for a programmable, portable data carrier the invention relates to the manipulation-safe creation of executable programme code for programmable portable data carriers, preferably in shape of smart cards.

From the US-A-6,023,565 is a method known to the distributed creation of a program for a programmable logic circuit. A user, who would like to prepare a programme for a such circuit by means of a computer located with it, becomes provided thereafter a simple operated user interface of the manufacturer of the circuits.

Thus the user on its computer describes the functionality desired for the logic circuit. The description made menu-guided over input masks, by means of those prepared parameters fixed become. The resultant, desired circuit functionality descriptive parameter data set becomes transmitted over a data network to a computer of the circuit manufacturer. This compiles the parameter data set and generated executable programme with the functionality desired of the user.

The executable programme sends the manufacturer back to the computer of the user, who it converts into a programming instruction sequence and transfers these to the logic circuit. As program preparation is reduced on dialogue-led entering of parameters, the possible concept also users without extensive programming knowledge the creation of programmes for logic circuits. A program preparation is possible thereby without the user has a compiler often commodity.

The concept turns off to handhave the application friendliness structurally heavy technical system to improve. Provisions to the protection of the data against manipulation, exchanged between the involved computers, are not met. The concept is not suitable therefore for applications, in which it particularly arrives on a protection of the generated program data against questioning intensively and manipulation. In particular it is not suitable in the described form the creation of programmes for smart cards, by means of those safety-relevant transactions, about banking transactions, executed to become to be supposed itself.

From US 6.005.942 a method is already to the safe introduction of an executable application on one in the field located smart card known.

The method possible it application offerers to bring to bottom engagement of the map publisher to arbitrary times during the life cycle of a smart card other applications on a card. Additional loadings of an executable application possible, those becomes the editor of the card associated is and the keys and cryptographic mechanisms the managed by means of a particular Kartendomain routine.

The Kartendomain routine becomes supported of safety routines, which likewise administer keys and cryptographic mechanisms, which are however the application offerer associated and secure applications which can be reloaded opposite the map publisher. Applications which can be reloaded are encrypted, become from the Kartendomain routine with support. the safety routines decrypted and into the card loaded.

With loadings the made examination of a cryptographic signature. With the creation of the applications on the basis of an application program source text, which can be reloaded, the writing does not deal.

From the WHERE 99/12307 is a method to the distribution of commercial software on the basis of a software producer over middlemen at user known. Described one will a method, it the middlemen allowed to add one software additional information which can be distributed without the security of the executable core code of the software which can be distributed becomes affected thereby. Achieved one becomes this by a particular dispatching routine, which provides one software encrypted which can be distributed and with a special distribution information table. Only in the latter the middlemen can make changes or attach additions.

Smart cards, which permit a reloading of executable programme code, as well as the introduction programme code in

smart cards, which can be reloaded, are z. B. in the " handbook of the smart cards " of W. Rankl, W. Effing, Hansa publishing house Munich, 3. Edition, described. Program preparation made thereafter complete on a background system. The created executable programme code becomes over one, z. B. by a mutual authentication saved interface on the smart card transfered. From safety reasons the made introduction of the executable programme code on the smart card preferably on-line, after an unique identification and allocation of background system, interfaces, map operating system and map microprocessor are made before. In order to ensure with keeping of a highest possible security the managableness that the identification informations contained databases of the background systems, the certificates to the creation of executable programme code on background systems of the map publishers only bottom editions issued and are listed the issued certificates. The possibility in principle created to provide executable programme code for smart cards themselves becomes limited thereby.

To the fuse of a data exchange between two computers, led across an open data network, a number of methods based on different coding techniques is known, and. A. the SSL (Secure Socket Layer) protocol, PGP (Pretty Good Privacy), the Secure Messaging or the SMI ME protocol. Methods of this type become also subsequent described in that, invention process used, are however actual not item of the invention. To the remark details including the kryptologischen implementation general is therefore referred to the various available descriptions of the respective methods in the pertinent literature as well as in the Internet. The same applies for used means to the usual in connection with backup processings as for instance the encryption in accordance with the 3DES-Verfahren or the formation of Message Authentication code (MAC).

The invention is the basis the object to indicate a method it with keeping maximum security against data manipulation allowed to permit to as large a circle of users as possible the creation of executable programmes for programmable portable data carriers. Object of the invention is it further to indicate the system components necessary to the carrying out the method.

The object becomes dissolved by a method with the features of the principal claim. According to invention a user a program editor to the creation a portable data carrier pre-mounted by program source texts as well as for the order provided, which software tools for finishing has, which the conversion of transportation code in executable programme code, present in a transition format, will permit. The creation of an executable program for the data carrier made distributed. With the program editor the created user a program source text, which becomes subsequent over a saved connection to a computer transmitted located with the editor of the data carrier. The saved connection can become thereby manufactured as a program source text of the pre-mounted data carrier becomes a transportation code encrypted and saved against change in such a way that only more certain, over with an editor of the data carrier located a computer of addressed receivers entschlüs seln and on integrity can examine the transportation code.

From the received program source text a generated programme code executable with the editor of the data carrier located computers by compiling and linking. Component Kompilier-und of link procedure is a formal verification of the generated programme codes, becomes determined by which in particular aggressive code. With the editor of the data carrier located computers convert the verified, executable programme code into a transition format and transmitted this over the computer of the user to the pre-mounted, portable data carrier. This transfered it with the help of the finishing often commodity tools again to executable programme code and transfers this to his memory.

Preferably the safety-relevant parts are contained of the finishing software in the pre-mounted data carrier. Thus purpose mA become ssig in the pre-mounted data carrier in particular the decryption and/or the detection of the authenticity and/or the integrity a programme code of a contained transportation code executed, before deposited in the correct case the resultant, executable programme code becomes into the memory of the data carrier.

The invention process creates safe " end end " a connection between computer located with an editor and a data carrier over a computer located with a user.

By the design of pre-mounting and the choice of the software tools it can be adapted thereby to easy to the type and the technical possibilities of the data carriers given in each case. The data carriers are only for the execution of symmetric coding procedures established, the made manufacturing saved " end end " of a connection convenient by use of a symmetric map-individual key on the one hand, and a superimposed, simplified asymmetric encryption on the data connection between the computer of the user and with the editor of the data carrier located computer on the other hand. In an alternative embodiment made to the fuse of the data transfer between the computer of the user and with the editor of the data carrier located computer an asymmetric encryption with mutual authentication and becomes the saved " end end " - connection between with the editor located computer and the data carrier with the mechanisms of the Secure Messaging established.

If the data carriers for the execution of asymmetric coding procedures are established, convenient between the " end end " saved direct with the editor of the data carrier located computer and the data carrier by asymmetric encryption becomes connection formed. The computer of the user functions thereby only as an intermediary.

The invention process has the advantage that the creation of executable programmes for a data carrier in principle to arbitrary users will leave can, without the identity of the user would have to become found and managed. Since the editor of the data carriers is included into each program preparation, the security of the generated programmes and over it the whole system is always ensured. Because in particular compiler functionality remains with the editor of the data carriers, important and safety-relevant know-how does not have to be handed over to the users.

By design of compiler functionality that direct, unencrypted accesses to program source texts derived of users or generated executable programme codes become blocked, reverse ensure leaves itself in such a manner that user-specific know-how of the users before the editor becomes protected. Convenient one will become for this in with the editor located computer a hardware security module used, in that compiler functionality, the en/decryption of programmes, the examination/creation of signatures as well as the authentication the executed. Outside of the hardware Sicherheitsmodules erschei program source texts or executable programme codes nen only in encrypted form.

By the formal verification of new created programmes with the editor, D. h. furthermore in safe environment, the very reliable introduction of aggressive programme codes can become into systems useful by means of a data carrier prevented. Besides the advantage results that all created executable programmes with the most current in each case compiler become compiled. The invention process can become thereby on-line or off-line executed. For the editors of data carriers the invention process opens even the possibility to deliver the creation of the desired in each case executable application programmes completely the users too left and the data carriers at all only in pre-mounted form. The integration of the editor into a program preparation possible furthermore the introduction of use methods, always forced by distributed program preparation, which use fee models, which z. B. on the number or the type of the executable programmes brought on a data carrier are based.

An embodiment of the invention becomes subsequent bottom reference on the drawing more near explained.

Show: Fig. 1 a system to the execution of a program preparation, Fig. 2 the structure of the integrated circuit of a programming of cash, portable data carrier, Fig. 3. the structure of a second computer, Fig. 4 the basic flow of a distributed Programmerstel lung, Fig. 5 to 7 flow charts to the illustration of the flow of one Program preparation, Fig. 8 the principle of an on-line examination of a created program on executability.

Fig. 1 the illustrated basic structure of a system to the distributed creation of a program for a programmable, portable data carrier. A first computer 20 formed for a data exchange with a portable data carrier 10 is 30 connected over a data connection 28 with a second computer.

The first computer 20 is with a user, approximately with a bank, an insurance, a retailer, a medical device or such or with a Dienstleister, that on behalf the aforementioned Eiririchtungen of programmes created. It possesses first, contacting or contactless working interface 24, the z. B. as Kontaktfeld, in form of a coil or as optical signal transmitters realized to be can do and a data exchange with a portable data carrier 10 the possible.

Over an other interface 26 it is 28 connected to a data connection. By both interfaces 24.26 the user computer 20 connects the data carrier 10 with the data connection 28. The user computer 20 makes 10 thereby auxiliary functions to the data carrier available. It allowed in particular the operation one, in the following short editor mentioned, editing program 22, that the creation of source texts of programmes for a data carrier 10 allowed.

For the programmable, portable data carrier 10 the subsequent form of a smart card is taken as a basis. On this aspect it is not however by any means limited. The data carrier 10 can be rather, adapted to the respective use, also differently formed, approximately in shape of a clock, than write means etc. Independent one of its concrete aspect possesses the portable data carriers 10 in each case an interface 14 corresponding to the interface 24 of the user computer 20, which a data exchange with a user computer 20 possible. Furthermore possesses the portable data carriers 10 an integrated circuit 12, which exhibits a central processing unit as well as a memory to the receptacle of the programme code at least one in the central processing unit of executable application program. The second computer 30 is typically with an editor of portable data carriers 10 or with an authorized operator of the here described method. Usually it possesses one compared with that user computer 20 and/or. the portable data carrier 10 essential larger arithmetic performance. The second computer 30 does not have to be thereby as structural unit realized. It can be rather also as system with distributed components executed, which over a particular data network connected is. To storage and/or. to the execution of sensitive functions hardware security modules can be used. Over an interface 34 the second computer is 30 28 connected to the data connection. The second computer 30 is in particular formed to implement a compiling program 310 for the conversion one in a programming high-level language of present source text program to machine language; it becomes therefore subsequent designated as compiler servers.

The data connection 28 has usually the shape of a data network and can by the Internet realized be in particular. Although in Fig. 1 only a connection between two components 20.30 shown is, can over in the following the data network data connection specified 28 also several user computers 20 with or also several compiler servers 30 connected be.

Fig. the structure of the integrated circuit 12 of a smart card 10 also software tools applied as pre-mounting shows 2. The integrated circuit 12 possesses an architecture typical for smart card processors and exhibits a central processing unit 100, a volatile working memory 102, as well as a non volatile memory array 104, latter existing from a non volatile read-only memory as well as a non volatile, rewritable memory. Usually the volatile working memories 102 a ram memory, the non volatile read-only memories is a ROM and the non volatile, rewritable memories an EEPROM memory. Except this mentioned arbitrary others can become, the same functionality exhibiting storage types used. Furthermore the central processing unit 100 is 14 connected with the interface.

In the non volatile memory array 104 is a number of for the use of the data carrier 10 required software tools, which become applied in a pre-mounting phase before transfer of the data carrier 10 to a user. Bottom software tools are not here all by a user variable programmes, routines or records understood to become, which are more insertable

bedarfsweise certain in each case tasks of data processing to the execution. In the frame of pre-mounting applied becomes thereby execution-independent, always similar map basic equipment 110. It covers at least the operating system 111, a basis program code 112 to the implementation, executable programme code reloaded by applications, which already are with transfer to the user on the smart card 10, as well as a storage area 113 to the later receptacle of.

On the other hand a selection of the subsequent software tools tuned on the selected in each case execution variant becomes applied: one for the integrated. Circuit 12, and thus for the smart card 10, individual and unique identification information 114, z. B. a serial number, a programme 116 cryptographic algorithms asymmetric to the execution, a programme 118 cryptographic algorithms symmetric for execution, a programme 120 to the guide of a data exchange after the principle of the Secure Messagings, a programme 122 for the execution of a data exchange over the data network 28 in accordance with the SSL protocol, an smart card-individual signature key 124, an smart card-individual symmetric key 126, the public key 128 one the smart card 10 of associated compiler server 30, a private map key 130 to the use in an asymmetric coding procedure, a certificate 132, which becomes the Zusammengehörigkeit between public map keys and identification informations with a signature of an editor confirmed, memory space the receptacle meeting keys of a 134-dieser in contrast to the aforementioned keys with the receptacle a data exchange with a compiler server 30 new in each case generated, as well as a sequence counter 136. All software tools mentioned can be also in each case multiple present. That particularly applies to the listed keys, certificates and the sequence counter.

Fig. 3 the illustrated structure of a compiler server 30 with the programmes used at the time of the execution of a program preparation and software tools. Core of the compiler server 30 forms a central processing unit 300, which 28 connected over an interface 34 with the data network is, in order to lead across it a data exchange with a user computer 20 and across it with a smart card 10. More other the central processing unit 300 a volatile working memory 302, usually in shape of a ram memory, is as well as a non volatile memory array 304 associated, which cover usually a only reader COM memory as well as a mass storage, about an hard disk.

In the memory array 304 are for the execution vorgeschlage of the nen method required software tools deposited. Fig. the simpleness because of an overview shows 3 over all software tools coming in connection with this description into considerations. The selection of the actual required software tools hangs, as with the smart card 10, of the selected embodiment concrete to the implementation of the method. General ones can be in the memory array 304 at software tools: , In Fig. 3 compiler compiling program mentioned 310 for the conversion of program source text to a programme code, in Fig. 3 linker link program mentioned 312 for the integration from already created programme codes into the context of a new created program, a code library of 318 programmes and program sections already present with the programme code, a database 320 programme codes associated to the tray of certain users, a Debug programme 316 for the examination of a created program on executability, a programme 321 for the formal verification of generated programmes and/or source texts, or several master key 324, which corresponds to that or the smart card-individual, symmetric keys 126, or several master keys 326, which to the smart card-individual keys 124 corresponds to MACs to the formation of data protection codes, in particular, or several public server keys 328 for the execution of asymmetric cryptographic algorithms, or several corresponding private server keys 330, or several public map key 332 algorithms, or several server certificates 334, asymmetric for execution, or several sequence counters 338, as well as a list with certificates, which became 10 formed with the manufacturing of the pre-mounted smart card and become 132 stored in the smart card 10 in the area. Furthermore the included memory array 304 a user list 340 with identification informations, which make an unique identification for a smart card possible; Identification informations to the identification of smart cards 10 can be for example their serial numbers.

Convenient ones become only the actual required established in a compiler server 30 with the practice of the method of the aforementioned software tools, not in each case the required omitted.

Meaning and use in the pre-mounted smart card of the 10 and/or. the compiler server 30 present software tools subsequent become on the basis the Fig. 4, which shows the basic flow of a distributed program preparation, as well as the Fig. 5 to 7 explained, which illustrate three embodiments of a distributed program preparation.

Fig. 4 shows first the basic flow of a distributed program preparation. In a preparatory phase a user a smart card 10 pre-mounted by applying software tools as well as an editor become 22 the order provided, step 400. With the editor 22 created it on the user computer 20 a program source text Q, step 402.

This with a transport lock, step 404, and into a transportation code T, TQ, TQssL transfered, step 406 is provided by application of an appropriate coding technology. The transportation code T, TQ, TQssL becomes 30 transmitted to the compiler server; Step 408.

The compiler server 30 waives the transport lock by decryption, step 410, and recovers in the transportation code the T, TQ, TQssL contained program source text Q, step 412. The program source text Q compiled, binds and for verified he subsequent, step 414. It results an executable programme code C, step 416, which becomes subsequent again transport-saved, to step 418. It becomes for this by application of appropriate coding mechanisms, which must agree not with 20 applied on pages of the user computer before, into a transition format U, UsM, USSL transfered, step 420. In this transition format it becomes 10 transmitted over the user computer 20 to the smart card, step 422.

Those determined using the software tools put on with pre-mounting from the transportation code U, USM, USSL by decryption, entered in the user computer 20, again the executable programme code C and finally loads this into his memory.

Fig. a distributed program preparation shows 5, 30 achieved with which data security becomes by use of on the smart card 10 prepared means in interaction with the compiler server. In Fig. 5 illustrated embodiment is particularly suitable for systems, in which the used smart cards control 10 only symmetric coding techniques.

Fig. an embodiment shows 6, with which the data transfer being made between user computers 20 and compiler server 30 by the data network 28 is by means of a SSL protocol saved, while that becomes direct between smart card 10 and compiler server 30 taking place data transfer in accordance with the Secure Messaging mechanism executed. The embodiment is suitable likewise for systems, in which the used smart cards permit 10 only symmetric coding techniques.

Fig. 7 an illustrated embodiment, with which the user computer works 20 essentially only as an intermediary between smart card 10 and compiler server 30. The fuse between smart card of the 10 and compiler server 30 transported data made, by between compiler server 30 and smart card 10 using the SSL protocol direct saved "end end" - connection established becomes. Table 1 illustrated the systematic applicability of the three subsequent on the basis the Fig. 5,6,7 described embodiments in response of the execution of the data transfer, the equipment requirements of the smart card 10 and the type of the transport lock.

Table 1
EMI16.1

<tb> Verfah Datenübertra request <September> on <September> Chipkar type <September> <September> Transport
<tb> ren <September> gung <September> width unit <September> Fuse
<tb> Fig <September> 5 <September> Off-line <September> Only <September> symmetric <September> Algo encryption
<tb> rithmen <September> and <September> MAC <SEPTEMBER> through
<tb> Chipk.
<tb>

Fig. <September> 6 <September> On-line <September> Symmetric <September> and/or <September> Secure <September> Messaging
<tb> asymmetric <September> Algorith through <September> Smart card
<tb> men
<tb> Fig. <September> 7 <September> On-line <September> Symmetric <September> and <September> Asym SSL <September> by <September> Chip
<tb> metrical <September> Algorithms <September> map
<tb> The left column in the Fig. 5,6,7 shows in each case the activities of the compiler server 30, the right activities of the user computer 20 and/or. the smart card 10, whereby " N " the user computer 20 designated, "K " the smart card 10.

In Fig. 5 represented program preparation upstream is a preparatory phase. Therein the user by the editor a pre-mounted smart card 10, step 500, as well as an editor becomes 22 the mechanism on its computer 20, step 502, provided. On the pre-mounted smart card 10 are and/or. 113 established are beside the basic equipment: an identification information ID in the storage area 114,
Programme 118 for the execution of symmetric cryptoalgorithms, for instance the " 3DES " - algorithm, at least an map-individual key KMAC to the formation of a data protection code, preferably in shape one
MACs, in the storage area 124, at least a key KENC to the symmetry schen encryption in the storage area 126, as well as memory location 134 to
Receptacle of at least two session keys SKENC, SKMAc. 10 at least two sequence counters 136 with values SEQc, SEQH are more other on the pre-mounted smart card. established. The sequence counter SEQc serves thereby for the calculation of the session keys SKENC, SKMAC for the safe over carrying of program source texts Q of the user computer 20 for the Compi more lerserver 30 used becomes, the sequence counter SEQH serves for the calculation of session keys SKENC, SKMAC, those for the safe transmission of
Programme codes C of the compiler server 30 to the user computer 20 used become.

On the compiler server 30 10 two become for each spent smart card
Sequence counter 338 with values the SEQc, SEQH established. The values SEQc, SEQH tune, thus the compiler server 30 after the same calculation specification, as them the smart card 10 begin, the session keys SKENC, SKMAc to compute can, always with the values of the corresponding
Sequence counter 136 of the associated smart card 10. The increase of the security become with each transmission of program source texts Q and/or. Programme code C other session keys SKENC, SKMAc used.

For this purpose smart card 10 and compiler servers increase 30 in each case before the calculation of meeting key SKENC, SKMAC the values SEQc and/or.

SEQH of the sequence counters 136,338.

The editor 22 the allowed creation of program source text Q, z. B. in a programming high-level language. Preferably supported it the Programmer position by a graphic underlaid, dialogue-controlled input guidance and offers direct useful development aids like a syntax check or the integration from program interfaces to the code library.

Are available smart card 10 in pre-mounted form and user computer 20, the created user using the editor 22 the program source text Q of a program certain to the introduction into a smart card 10, step 504. In addition, the preferably made creation in a programming high-level language, general is every other format possible.

If a program source text is Q created, the user assigns the smart card 10 over the editor 22 by means of a corresponding instruction to code and secure with a MAC against change the program source text Q. In addition the increased smart card 10 first the sequence counter value SEQc and the generated meeting keys SKENC and SKMAc, z. B. with the symmetric 3DES-Algorithmus, step 506. Subsequent encrypted it the program source text Q with the session key SKENC to an intermediate code Q'und calculated with the session key, obtained of the editor 22 over the interfaces 24.14, SKMAC a MAC over Q', step 508. Intermediate code Q'und MAC hands the smart card to 10 then over the interfaces 24.14 back to the editor 22 over.

This determined furthermore the map identification ID, put on in the storage area 114 of the smart card 10, step 509, and join it with the intermediate code Q as well as the MAC to a transportation code T. Such a formed transportation code T the transmitted user computer 20 over the data network 28 to the compiler server 30, step 510. The transmission of the transportation code T can be made thereby in arbitrary manner by an unsecured medium. For example the transportation code T can become as E-Mail transfered or on disk by posts to the editor sent. Since beside the transportation code T can become also on-line over the data network 28 to the compiler server 30 sent. Privacy and integrity of the transmitted transportation code T become 10 ensured by the encryption and the MAC calculation by the smart card.

With the compiler server 30 received, this examines first, step 512 whether the identification information ID contained in the transportation code T is contained also in the identification list 340 led in the compiler server 30, which preferably forms a customer master list. Applies, it derives the associated map-individual keys KENC and KMAC first from the master codes MKENc and MKMAc located in the storage areas 324.326 with the help of the identification information ID, step 514. From these keys as well as the incremented sequence counter SEQc the calculated compiler server 30 then the session keys SKENC and SKMAc after the same calculation specification, which has before the smart card 10 used. With the session key SKMAC the calculated compiler server 30 subsequent for his part one MAC', step 516, and compares it with the MAC contained in the transportation code T. , The compiler server 30 recognizes voices both the transportation code T as authentic, D. h. coming from the smart card 10 with the identification information ID, and more integer, D. h. not with the transmission changed.

The compiler server 30 has a transportation code T as authentic recognized, decrypted it the program source text Q' contained in the transportation code T by means of the session key SKENC. Due to the integrity of the transportation code determined before T the resultant, decrypted format agrees with the created program source text Q original on the user computer 20.

The restored program source text Q the transfered compiler he 30 using the compiling program 310 into an intermediate format, which it connects to subsequent by means of the link program 312 bottom accesses to the code library 318 with already present programme code, step 518.

Compiling program 310 and link program 312 are in a convenient design in form of a hardware Sicherheitsmodules executed, which Compilier-und link functionality, Ent and the encryption of the processed program data, the examination and creation of signatures as well as the authentication included. All processed program data, in particular incoming program source texts Q and generated executable programme codes C appear then outside of hardware the Sicherheitsmodules only in encrypted form. On these ensure leaves itself that user-specific know-how of the users becomes 30 protected against insight and access over the compiler server.

Convenient one can be in the compiler servers 30 furthermore a limitation of the access to the code library 318 established, which z. B. the integration already present programme code in new generated by the link program 312 limited.

The programme code C resultant after compiling and linking becomes 321 formal verified by means of the verification program. The programme code becomes C on obvious errors tested, approximately on adherence to the address space, on attention of the predetermined memory sizes, on type injuries or on aggressiveness, step 520.

The programme code C generated from the program source text Q is usable thereafter, D. h. by the smart card 10 more executable, he becomes for the retransmission into a transportation code U converted. For this first the sequence counter value becomes SEQH increased. With the increased sequence counter value SEQH subsequent from the master codes MKENc becomes and/or. MKMAC and the identification information ID the map-individual keys KENC and KMAC derived and thus again session keys SKENc and SKMAc calculated, step 522. The calculation of the session keys SKENc, SKMAc by the compiler server 30 made on the same manner as it before, in step 506, of the user computer 20 made became, whereby only in place of the sequence counter value SEQc becomes the sequence counter value SEQH used.

Subsequent one becomes with the session key SKENc the programme code C an intermediate code C'verschlüsselt and over the intermediate code C'mittels session keys of the SKMACweiterhin a MAC calculated, step 524.

Intermediate code C'und MAC then assembled, which the compiler server 30 to the user computer 20 sends, step 526 becomes a transportation code U. For the transmittal of the transportation code U as in case of the transportation code T

any can become, in particular also an actual uncertain transmission medium like a disk or the shipment by enamel selected. Of course 28 possible beside it also the use of an on-line connection is over a data network. If an on-line connection becomes used, the possibility, to a job, exists D. h. the sending off in a transportation code contained of a program source text Q to a compiler 30, in a single on-line session also the result, D. h. a transportation code U with the programme code C'zu obtained.

The user computer 20 leads the obtained transportation code U over the interfaces 24.14 other to the smart card 10, step 528. That one the increased value SEQH of the sequence counter 136, generated thereby on the same manner as before the compiler server 30 in step 522 the meeting keys SKENc and/or. SKMAC and examines whether the MAC transmitted with transportation code U is the " identical MAC, which the smart card 10 even by means of the key SKMAC from U can compute step 530. Voices MAC " and MAC, is the MAC " from U successful verified. Since except the smart card 10 even only the compiler server 30 has the possibility, the key sports club MAC to use, is by decryption of the programme code transmitted in the transportation code U C'gewonnenen programme code C authentic, D. h. it became of the compiler server 30 out a program source text Q generated transport-saved of the same smart card 10. As authentic recognized programme code C becomes 113 loaded of the smart card 10 into the map memory, step 532.

Fig. an embodiment of a distributed program preparation shows 6 as flow chart, with which the data transfer being made between user computers 20 and compiler server 30 by the data network 28 is by means of SSL saved, while the direct data transfer between smart card 10 and compiler server becomes 30 executed with the help of the Secure Messaging mechanism. The embodiment is suitable as in Fig. 5 illustrated embodiment particularly for an on-line execution in systems, in which the used smart cards permit 10 only symmetric coding techniques.

One for the execution of the second embodiment pre-mounted smart card 10 covers 110 with operating system 111, basis program code 112 and memory space 113 for completion program code, beside the basic equipment a routine 120 for the execution of the Secure Messagings, a private map key 130 as well as a public server key 128. Furthermore the user computer 20 has program functionality to the execution of the SSL protocol without authentication of smart cards.

The execution of a program preparation in the second embodiment corresponds first to the first embodiment in accordance with Fig. 5 and covers the steps 500 to 504.

If a program source text Q is present, the user furnishes a connection between its computer 20 and the compiler server 30 of the editor, step 600 over the data network 28.

If the physical connection is 30 manufactured to the compiler server, a SSL protocol started becomes between user computers 20 and compiler server 30. User computer 20 and compiler server 30 determine thereby a session key SKssL, steps 601,602 in each case. Subsequent one becomes within the SSL protocol a so called IP tunnel between compiler server 30 and smart card 10 the execution of the Secure Messagings established, step 604. In the user computer 20 thereby the protocol of the Secure Messagings accomplished by the smart card 10 becomes into that, only between user computer 20 and compiler server 30 used SSL protocol embedded. In the IP tunnel subsequent smart card-specific records become, preferably in shape of APDUs (Application Protocol DATA unit) direct between smart card 10 and compiler server 30 transported. Regarding the Secure Messaging the user computer 20 functions only as pure intermediaries.

In accordance with the Secure Messaging smart card 10 and compiler server 30 accomplish then a mutual authentication, whereby first the card 10 opposite the compiler server 30 authenticates itself, step 606, the subsequent compiler server 30 opposite the card 10, step 608. If the mutual authentication between smart card 10 and compiler server 30 successful runs, the use of all functions of the compiler server becomes 30 20 enabled by means of the user computer, step 610.

The use release is present, the encrypted user computer 20 the created program source text Q with the before certain session key SKsEs and the transmitted transportation code TQ resultant from it to the compiler server 30, step 612.

In the compiler server 30 the transportation code TQ is entered with the help of before in the compiler server 30 generated session key SKssL again decrypted, step 614, and into the source text Q transfered created on the user computer 20. Convenient one the made execution of the steps 610, 612.614 in form of a continuous data exchange between user computer 20 and compiler server 30, so that the recovery of the source text Q in the compiler server becomes 30 immediate after receipts of the last encrypted source text data record of the user computer 20 completed.

From the source text Q the generated compiler server 30 then by execution on the basis the Fig. 5 described steps 518 and 520 an executable programme code C.

The compiler server 30 changes the executable programme code C by application of the Secure Messaging mechanisms in saved programme code CsM, step 620. The saved programme code CSM transfered it subsequent by encryption with the help of session keys the SKsEs into a transportation code UCsM, step 622 present in a transition format. By the encryption with the session key SKsEs becomes that, typically in shape of APDUs present, saved programme code CsM into a fuse of the data transfer over the data network 28 between compiler servers 30 and user computer 20 embedded.

The transportation code UCsM the transmitted compiler server 30 to the user computer, present in the transition format, 20. This decrypted UCsM by means of the meeting key SKsEs, step 626, whereby the fuse mounted to the protection of the data transfer between compiler server 30 and user computer 20 becomes again remote. The decrypted programme

code CsM saved present thereafter in accordance with the Secure Messaging the user computer hands to 20 to the smart card 10 over, step 624.

In the smart card 10 the saved programme code CsM becomes by application of the turning around Secure Messaging mechanisms again in executable programme code C returned, step 628, and finally into the memory array 104 into the area 113 loaded, step 630 there prepared to the receptacle of completion program code.

From reasons of the clarity became managing in Fig. 6 represented procedure as sequential sequence of separate steps described. In the practice between smart card the 10 usually contains, user computer 20 and compiler server 30 taking place data transfers a data exchange in in each case two directions. Meaningful one is it besides to implement method steps, for which the possible is, in form continuous, quasi parallel Datenaustausch-und of processing process in the compiler server 30 and user computer 20 and/or. Smart card of 10 method steps. temporal overlaying implement. Convenient one is this z. B. for the steps 620 to 630: they become preferably in shape of a continuous data exchange between compiler server 30 and user computer 20 executed, in which a transmission of records of the transportation code UCsM already takes place to the user computer 20, during on the compiler server 30 still the conversion of the programme code C in accordance with the Secure Messaging made, and in that 10 records transfered of the compiler server 30 over the user computer the 20 to the smart card by these immediate before that loadings into the memory space 113, D. h. without intermediate storage up to the complete input, decrypted become.

Fig. 7 an illustrated other embodiment on the basis the Fig. 4 described program preparation, with which the user computer works 20 essentially only as an intermediary between smart card 10 and compiler server 30. The fuse between smart card of the 10 and compiler server 30 transported data made, by between compiler server 30 and smart card 10 using the SSL protocol saved, direct " end end " - connection established becomes.

Pre-mounting a smart card 10 included appropriate for the execution of this execution variant beside the mechanism of the basic equipment 110 with operating system 111, basis program code 112 and storage area for completion program code 113 the application of a program 122 for the execution of the SSL protocol, the deposit of a certificate 132, the deposit of the private map key 130 as well as the deposit of the public server key 128.

The carrying out the method in accordance with Fig. to 7 corresponds first on the basis the Fig. 5 described embodiment and covers the steps 500 to 504. They become followed of the mechanism of a connection between the smart card 10 and a compiler server 30 over the user computer 20, step 700.

Smart card 10 and compiler server 30 implement now a complete SSL protocol. Within the handshake procedure made here a mutual authentication, as the compiler server certificate becomes 332 10 tested by the smart card, step 701, on the other hand the certificate 132 by the compiler server, put on in the smart card 10, 30, step 702. If a continuation of the data exchange is possible after the mutual certificate examination, smart card 10 and compiler server 30 generate in each case a session key, step 704 and/or. 706.

In Fig. 7 illustrated embodiment is particularly suitable for on-line execution. After manufacturing of a safe data connection between smart card 10 and compiler server 30 therefore provided can be that the user must meet a bottom selection several possible operating options for further processing a selection. In this case the compiler server 30 sends a report of offering to manufacturing of the safe data connection over the possible operating options to the user computer 20, step 708. From the communicated options the user selects the desired, about a program preparation with on-line translation, step 710 over the user computer 20, or a Debug mode, becomes on-line found in which the feasibility of a new generated programme code.

To the increase of the security of the data transfer to the compiler server 30 subsequent can be an optional signature of the program source text Q by the smart card 10 provided, step 711. The signature made in actual known manner, as the smart card forms 10 over the source text Q an hash worth and this with the private key 130 of the smart card encrypted. The Hashwertbildung can take place thereby, in particular with not sufficient hardware resources on a smart card 10, by the user computer 20.

That, if necessary program source text code the encrypted smart card 10 with the before certain session key SKssL mark to a transportation code TQssL, step 712, which them send to subsequent over the user computer 20 to the compiler server 30, step 714.

That one the decrypted received transportation code TQssL again with the session key SKsEs, step 716, in order to restore the program source text Q. Case a signature present is, examines it by renewed formation of the Hashwertes using the public map key 332 their correctness.

From the restored program source text Q the generated compiler server 30 subsequent by execution of the steps 518.520 an executable programme code C.

The compiler server 30 provides the generated programme code C with a signature, which it by formation of a Hashwertes and coding the Hashwertes with the private key 330 of the compiler server 30 generated.

The developed, signed code encrypted it then with the public key 332 of the smart card 10, step 718. Thereafter present the Chiffrat the transfered compiler server 30 subsequent by coding with the session key SKsEs into a transition format CssL, step 720, which it as transportation code finally to the user computer 20 transmitted, step 722.

This leads the received transportation code CssL to the smart card 10 more other, step 724 which from it by decryption with the session key SKsEs again the Chiffrat of the executable programme code generated, step 725. Case the programme code C in the compiler server 30 signed became, the decrypted smart card 10 the Chiffrat other with the private key 130 of the smart card 10 and examines the signature with the public key 128 of the compiler server 30, step 726, present thereafter. If the result of the signature examination is positive, the smart card 10 loads the thus present executable programme code C into the Speicheranordung 104 into the memory space 113, step 728 planned to the receptacle of completion program code.

As is the case for the embodiment after Fig. 5 became in Fig. 7 represented procedure of the clarity because of sequentially described. Practical one is it however meaningful, method steps, for which the possible is, quasi parallel to implement, as compiler server 30 and smart card implement 10 it temporal overlaying. That applies for z. B. for the steps 712 to 716, D. h. the smart card-lateral encryption and the compiler-server-lateral recovery of the program source text Q. They take place convenient in form continuous, quasi parallel Datenaustausch-und of processing process, so that the program source text Q almost immediate is present after sending off of the last record by the smart card 10 in the compiler server 30. An intermediate storage up to the complete input of the program pouring text Q made not. An implementation in shape continuous, quasi parallel Datenaustausch-und of processing process continues to itself offer 718 to 728 also for the steps, D. h. for the compiler-server-lateral encryption of the executable programme code C and its smart card-lateral recovery as well as loadings into the memory space 113 on the smart card 10. The execution of these steps by compiler server 30 and smart card 10 made convenient without intermediate storage more immediate datasentence by sentence, so that the executable programme code C essentially immediate is present after sending off of the last transportation code data record by the compiler server 30 in the memory of the smart card 10.

In the frame of a program preparation in accordance with one that managing described embodiments can the execution of a Debug routine provided be. Thus a programme code C before that, created by the compiler server 30, becomes loadings on a smart card 10 on executability tested. The principle of such a Debug routine is in Fig. 8 illustrated, whereby to the simplification of the description the measures directed to the fuse of the data transfer, D. h. above all the various encryptions, not shown are.

The Debug routine is 30 applied as programme 316 in the compiler server and becomes also there executed. Additional or as component of the program 316 included it a data carrier copying hardware to the simulation and/or a software related reproduction of a data carrier for the emulation of a generated program on the compiler server 30 the bottom technical boundary conditions present on the data carrier. Controlled one becomes it, after setting of a corresponding mode of operation in the compiler server 30, over the editor 22 in the user computer 20. The operating mode attitude knows z. B. in the frame of the selection of an operating option in the steps 708 and 710 take place, if in accordance with program preparation in Fig. 6 illustrated embodiment made becomes. The Debug mode of operation allowed it and. A. to start from the user computer 20 from a programme generated in the compiler server 30 to set stop marks storage areas to indicate as well as variables select and set.

For the execution of the Debug routine first a program source text Q created, step 504, and a connection becomes the compiler server 30 constructed, step 700 in conventional manner. Subsequent one becomes the source text Q in accordance with one of the before described embodiments to the compiler server 30 transmitted, step 800.

If the source text Q was received, the compiler server 30 offers the generation of a programme code C in the Debug mode of operation to the user, step 802. A user can select the mode on it over the user computer 20, step 804. Became the Debug mode of operation selected, the created compiler server 30 from the received program source text Q by execution of the steps 526.528 a preliminary programme code Cv, which is executable on Simulations-und present in the compiler server 30/or emulation environment. The compiler server 30 stores the preliminary programme code Cv into a buffer, step 806. Subsequent transmitted it the user computer 20 a production message, step 808, which brings these to the display, step 810.

The user can provide the source text program Q now by means of the user computer 20 with Debug instructions, D. h. Stop marks set, the execution of a program in single steps or displays of variables arrange, step 812. The Debug instructions become the compiler server 30 reported.

Subsequent one can become over the user computer 20 the execution of the program on the compiler server 30 triggered, realized by the preliminary programme code Cv, step 814. The compiler server 30 implements on it the programme bottom consideration of the Debug instructions communicated before, step 816. In each case after execution of a program section transmitted it specified by the Debug instructions a result message to the user computer 20, step 818, which brings these to the display, step 820. Dependent of the handed over Debug instructions on it an intervention of a user can be into the programme execution provided, approximately by input of variables or by setting new Debug instructions, step 822. If necessary made interventions into the program source text Q or new Debug instructions, the transmitted user computer 20 the compiler server 30. A Debug instruction is finally processed, the transmitted user computer 20 the compiler server 30 a continuation signal, step 824, on that that by repetition of the step 814 the execution of the next program section caused. Considered it eventual made interventions into the program source text Q or new Debug instructions. The steps 814 to 824 become repeated, until the compiler server 30 has a programme complete executed realized by a preliminary programme code Cv.

The programme proves finally as correct executably, to the caused user over the user computer 20 a change of the mode of operation into the standard mode, step 826. The compiler server 30 generated thereupon from at this time the present program source text Q an executable programme code C and transfers this as on the basis the Fig. 4 to 6 described over the user computer 20 to the smart card 10, step 828.

Furthermore it deletes the latched, preliminary programme code Cv, step 830 those managing described sequence about embodiments is as basis for a concrete procedure realization to be understood in each case. Bottom Beibehal tung the basic approach to use for the achievement of a safe program preparation pre-mounted data carriers the embodiments are in each case in a far frame out shapable. This applies in particular to the execution of the structural elements, D. h. the smart card, the user computer, the data network and the compiler server. More other Verschlüsselungs-und mentioned Authentifizierungsverfahren can become natural by other with same safety effect replaced.

All described embodiments can be brought in particular by the use of other keys, sequence counters or other kryptografischer algorithms on an again increased safety stage. From technical or from safety reasons also other Umformatie can be rungen provided. So it is in particular with programmable smart cards with view on their limited memory location usual to reformat an executable programme code generated on a compiler server 30 again before or with loadings into the smart card memory-optimizing as for example symbolic correlations become replaced by absolute addresses. Furthermore from reasons of clarity only in each case cases of good became described. The treatment of cases of an error can be derived from it however using known standard solutions.

Europäisches Patentamt
European Patent Office
Office européen des brevets

Claims of WO02069118	Print	Copy	Contact Us	Close

Result Page

Notice: This translation is produced by an automated process; it is intended only to make the technical content of the original document sufficiently clear in the target language. This service is not a replacement for professional translation services. The esp@cenet® Terms and Conditions of use are also applicable to the use of the translation tool and the results derived therefrom.

Claims 1. Method to the distributed creation of an executable program for a programmable, portable data carrier, whereby the creation of a program source text on first, with a user befindli chen computers, compiling and linking of the program source text to an executable programme code after transmission on two ten, with the editor of the data carrier located computer, and that
Gekenn it draws loadings of the executable programme code into the data carrier after back transmission again over the first computer made, thus that in a pre-mounting step on the data carrier (10) software tools become the finishing applied, which permits it, from egg nem transportation code present in a transition format (U, UCsM,
CssL to win) an executable programme code (C) in the second computer (30) generated, executable programme code (C) for the retransmission in transportation code (U, UCsM, CssL,) converted becomes, and to the first computer (20) to the introduction into the data carrier (10) returned transportation code (U, UCsM, CssL,) is reconverted by means of the software of tools in executable programme code (C).

2. Process according to claim 1, characterised in that the program source text (Q) for the transmission to the second computer (30), created on the first computer (20), a transport lock receives, by allotting pro rata becomes. 3. Process according to claim 1, characterised in that the software tools a data carrier (10) characteristic identification information (114) as well as a signature key (124) contain.

4. Process according to claim 1, characterised in that the software tools a programme for the execution SSL handshake of a protocol (122) and/or a programme for the execution of Secure Messaging (120) contain.

top 5. Process according to claim 1, characterised in that the software tools a private data medium key (130) as well as a programme for the examination of a signature with the public key (128) of a second computer (30) contain.

6. Process according to claim 1, characterised in that into the portable data carrier (10) loading, executable programme code (C) including the second computer (30) executed will, in order to locate possible errors.

7. Process according to claim 6, characterised in that into the portable data carrier (10) loading, executable programme code (C) on the second computer (30) including the first computer (20) executed becomes.

8. Process according to claim 1, characterised in that the executable programme code (C) in the memory (113) of the data carrier (10) deposited becomes.

9. Programmable portable data carrier with an integrated circuit, which exhibits a processor as well as a memory to the receptacle of programme code executable by the processor, characterised in that in the integrated circuit (are to be transferred of 12) software tools for finishing applied, which make it possible, a transportation code supplied in a transition format (U, UCsM, CssL) in executable programme code (C).

10. Data carrier according to claim 9, characterised in that it at least one sequence counter (136) exhibits.

11. Data carrier according to claim 9, characterised in that it the data carrier (10) characteristic identification information (114) as well as a key (124) to the formation of a data protection code contains, which assign it to a defined second computer (30).

12. Computer for the execution of a distributed creation of an executable program for a programmable, portable data carrier, contained at least a compiling program as well as a link program (312), characterised in that it over means orders, in order to recover out a happened transportation code present in a transition format (T, TQ, TQssL,) a program source text (Q).

13. Computer according to claim 12, characterised in that it means exhibits, in order to determine and examine the identity one data carrier (10), which can be programmed.

14. Computer according to claim 12, characterised in that it a table (340) leads, in for each portable data carrier (10), which by means of the computer (30) is more programmable, the data carrier characteristic identification information (114) deposited is.

15. Computer according to claim 12, characterised in that it over means (321) for the formal verification of a programme code (C) generated by compiling exhibits.

16. Computer according to claim 12, characterised in that it over means (316) orders, in order to examine a generated programme code (C) by immediate execution for whether it contains errors.

17. Computer for the execution of a distributed creation of an executable program for a programmable, portable data carrier, contained at least a first interface for a data exchange with a data carrier as well as a second interface to a data connection, characterised in that it means exhibits to make possible in order to implement an editing program (22) to the creation one by the computer (20) even not executable program source text (Q), and that it exhibits means, in order over first and the second interface (24,26) a direct data transfer between portable data more inertial (10) and one, a second computer (39), connected over the data connection (28).

18. Computer according to claim 17, characterised in that it means exhibits, in order to implement a SSL protocol.

19. System to the distributed creation of an executable program for egg nen programmable, portable data carrier, containing an inertial cash data carrier in accordance with according to claim 9 as well as a computer
Claim 12.

20. System of claim 20, characterised in that it further egg nen computers according to claim 17 enclosure.

▲ top

(12) NACH DEM VERTRAG ÜBER DIE INTERNATIONALE ZUSAMMENARBEIT AUF DEM GEBIET DES PATENTWESENS (PCT) VERÖFFENTLICHTE INTERNATIONALE ANMELDUNG

(19) Weltorganisation für geistiges Eigentum
Internationales Büro

(43) Internationales Veröffentlichungsdatum
6. September 2002 (06.09.2002)

PCT

(10) Internationale Veröffentlichungsnummer
WO 02/069118 A2

(51) Internationale Patentklassifikation[7]: G06F 1/00

(21) Internationales Aktenzeichen: PCT/EP02/01655

(22) Internationales Anmeldedatum:
15. Februar 2002 (15.02.2002)

(25) Einreichungssprache: Deutsch

(26) Veröffentlichungssprache: Deutsch

(30) Angaben zur Priorität:
101 08 487.0 22. Februar 2001 (22.02.2001) DE

(71) Anmelder *(für alle Bestimmungsstaaten mit Ausnahme von US)*: **GIESECKE & DEVRIENT GMBH** [DE/DE]; Prinzregentenstraße 159, 81677 München (DE).

(72) Erfinder; und

(75) Erfinder/Anmelder *(nur für US)*: **GOLLNER, Michael** [DE/DE]; Schwandorferstraße 3, 81549 München (DE). **CIESINGER, Daniel** [DE/DE]; Lily-Braun-Weg 15, 80637 München (DE).

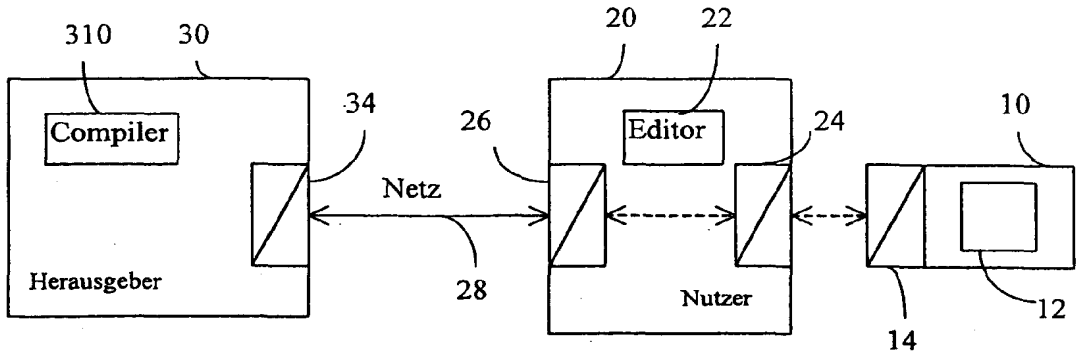
(74) Anwalt: **KLUNKER, SCHMITT-NILSON, HIRSCH**; Winzererstraße 106, 80797 München (DE).

(81) Bestimmungsstaaten *(national)*: AE, AG, AL, AM, AT, AU, AZ, BA, BB, BG, BR, BY, BZ, CA, CH, CN, CO, CR, CU, CZ, DK, DM, DZ, EC, EE, ES, FI, GB, GD, GE, GH, GM, HR, HU, ID, IL, IN, IS, JP, KE, KG, KP, KR, KZ, LC, LK, LR, LS, LT, LU, LV, MA, MD, MG, MK, MN, MW, MX, MZ, NO, NZ, OM, PH, PL, PT, RO, RU, SD, SE, SG, SI, SK, SL, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VN, YU, ZA, ZM, ZW.

[Fortsetzung auf der nächsten Seite]

(54) **Title:** METHOD AND SYSTEM FOR THE DISTRIBUTED CREATION OF A PROGRAM FOR A PROGRAMMABLE PORTABLE DATA CARRIER

(54) **Bezeichnung:** VERFAHREN UND SYSTEM ZUR VERTEILTEN ERSTELLUNG EINES PROGRAMMS FÜR EINEN PROGRAMMIERBAREN, TRAGBAREN DATENTRÄGER



310...COMPILER
30...PUBLISHER
28...NETWORK
22...EDITOR
20...USER

(57) **Abstract:** The invention relates to a method for the distributed creation of a program for a programmable portable data carrier (10), for example, a chip card. To this end, program source text (Q) is created on a user computer (20), compiled and linked to executable program code (C) on a spatially separate compiler server (30), and the executable program code (C) is loaded into the data carrier (10) once again via the user computer (20). A secure end-to-end link is established for conducting an exchange of data between the data carrier (10) and the compiler server (30). To this end, the data carrier (10) is provided, in a pre-completion step, with software tools for final processing, which permit a transport code (U, C_{ssl}, UC_{SM}) provided in a transition format to be converted into executable program code (C). The transport code (U, C_{ssl}, UC_{SM}) is secured by encoding mechanisms. The transmission of the executable program code (C), which is generated by the compiler server (30), ensues in the transition format (U, C_{ssl}, UC_{SM}).

[Fortsetzung auf der nächsten Seite]

WO 02/069118 A2

(84) Bestimmungsstaaten *(regional)*: ARIPO-Patent (GH, GM, KE, LS, MW, MZ, SD, SL, SZ, TZ, UG, ZM, ZW), eurasisches Patent (AM, AZ, BY, KG, KZ, MD, RU, TJ, TM), europäisches Patent (AT, BE, CH, CY, DE, DK, ES, FI, FR, GB, GR, IE, IT, LU, MC, NL, PT, SE, TR), OAPI-Patent (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, ML, MR, NE, SN, TD, TG).

Veröffentlicht:

— *ohne internationalen Recherchenbericht und erneut zu veröffentlichen nach Erhalt des Berichts*

Zur Erklärung der Zweibuchstaben-Codes und der anderen Abkürzungen wird auf die Erklärungen ("Guidance Notes on Codes and Abbreviations") am Anfang jeder regulären Ausgabe der PCT-Gazette verwiesen.

(57) Zusammenfassung: Vorgeschlagen wird ein Verfahren zur verteilten Erstellung eines Programmes für eine programmierbaren, tragbaren Datenträger (10), z.B. einer Chipkarte. Dabei erfolgen die Erstellung des Programmquelltextes (Q) auf einem Nutzercomputer (20), Compilierung und linken zu ausführbarem Programmcode (C) auf einem räumlich getrennten Compilerserver (30) und das Laden des ausführbaren Programmcodes (C) in den Datenträger (10) wieder über den Nutzercomputer (20). Für den Datenaustausch zwischen Datenträger (10) und Compilerserver (30) wird eine sichere Ende-zu-Ende Verbindung aufgebaut. Der Datenträger (10) wird dazu in einem Vorkomplettierungsschritt mit Softwarewerkzeugen zur Endbearbeitung ausgestattet, die es erlauben, einen im einem Übergangsformat vorliegenden Transportcode (U, C_{ssl}, UC_{SM}) in ausführbaren Programmcode (C) zu wandeln. Der Transportcode (U, C_{ssl}, UC_{SM}) ist durch Verschlüsselungsmechanismen gesichert. Die Übertragung des durch den Compilerserver (30) erzeugten, ausführbaren Programmcodes (C) an den Datenträger (10) erfolgt im Übergangsformat (U, C_{ssl}, UC_{SM}).

Verfahren und System zur verteilten Erstellung eines Programms für einen programmierbaren, tragbaren Datenträger

Die Erfindung betrifft die manipulationssichere Erstellung von ausführbarem Programmcode für programmierbare tragbare Datenträger, vorzugsweise in Gestalt von Chipkarten.

Aus der US-A-6,023,565 ist ein Verfahren zur verteilten Erstellung eines Programmes für einen programmierbaren Logikschaltkreis bekannt. Einem Nutzer, der mittels eines bei ihm befindlichen Computers ein Programm für einen derartigen Schaltkreis erstellen möchte, wird danach vom Hersteller der Schaltkreise eine einfach bedienbare Nutzerschnittstelle bereitgestellt. Damit beschreibt der Nutzer auf seinem Computer die für den Logikschaltkreis gewünschte Funktionalität. Die Beschreibung erfolgt menügeführt über Eingabemasken, mittels derer vorbereitete Parameter festgelegt werden. Der resultierende, die gewünschte Schaltkreisfunktionalität beschreibende Parameterdatensatz wird über ein Datennetz an einen Computer des Schaltkreisherstellers gesandt. Dieser compiliert den Parameterdatensatz und erzeugt ein lauffähiges Programm mit der vom Nutzer gewünschten Funktionalität. Das lauffähige Programm sendet der Hersteller zurück an den Computer des Nutzers, welcher es in eine Programmierungsbefehlsfolge umsetzt und diese an den Logikschaltkreis überträgt. Indem die Programmerstellung auf das dialoggeführte Eingeben von Parametern reduziert ist, ermöglicht das Konzept auch Nutzern ohne weitreichende Programmierkenntnisse die Erstellung von Programmen für Logikschaltkreise. Eine Programmerstellung ist dabei möglich, ohne daß der Nutzer über eine Compilersoftware verfügt. Das Konzept stellt darauf ab, die Anwendungsfreundlichkeit eines strukturbedingt schwer handzuhabenden technischen Systems zu verbessern. Vorkehrungen zum Schutz der zwischen den beteiligten Computern ausgetauschten Daten gegen Manipulation werden nicht getroffen. Das Konzept

eignet sich deshalb nicht für Anwendungen, in denen es besonders auf einen Schutz der erzeugten Programmdaten gegen Ausforschung und Manipulation ankommt. Insbesondere eignet es sich in der beschriebenen Form nicht zur Erstellung von Programmen für Chipkarten, mittels derer sicherheitsrelevante Transaktionen, etwa Bankgeschäfte, ausgeführt werden sollen.

Aus der US 6,005,942 ist ein Verfahren zum sicheren Einbringen einer lauffähigen Applikation auf eine bereits im Feld befindliche Chipkarte bekannt. Das Verfahren ermöglicht es Applikationsanbietern, unter Einschaltung des Kartenherausgebers zu beliebigen Zeitpunkten während des Lebenszyklus einer Chipkarte weitere Applikationen auf eine Karte zu bringen. Das nachträgliche Laden einer lauffähigen Applikation wird mittels einer speziellen Kartendomain-Routine ermöglicht, die dem Herausgeber der Karte zugeordnet ist und die Schlüssel und kryptographische Mechanismen verwaltet. Die Kartendomain-Routine wird unterstützt von Sicherheitsroutinen, die ebenfalls Schlüssel und kryptographische Mechanismen verwalten, welche aber dem Applikationsanbieter zugeordnet sind und nachzuladende Applikationen gegenüber dem Kartenherausgeber sichern. Nachzuladende Applikationen sind verschlüsselt, werden von der Kartendomain-Routine mit Unterstützung.der Sicherheitsroutinen entschlüsselt und in die Karte geladen. Beim Laden erfolgt die Prüfung einer kryptographischen Signatur. Auf die Erstellung der nachzuladenden Applikationen ausgehend von einem Applikationsprogrammquelltext geht die Schrift nicht ein.

Aus der WO 99/12307 ist ein Verfahren zur Verteilung von kommerzieller Software ausgehend von einem Softwarehersteller über Zwischenhändler an Endabnehmer bekannt. Beschrieben wird eine Methode, die es den Zwischenhändlern erlaubt, einer zu verteilenden Software Zusatzinformationen hinzuzufügen, ohne daß die Sicherheit des ausführbaren Kerncodes der zu

verteilenden Software dabei beeinträchtigt wird. Erreicht wird dies durch eine spezielle Versenderoutine, die eine zu verteilende Software verschlüsselt und mit einer besonderen Verteilungsinformationstabelle versieht. Nur in der letzteren können die Zwischenhändler Änderungen vornehmen oder Ergänzungen anbringen.

Chipkarten, die das Nachladen von ausführbaren Programmcode erlauben, sowie die Einbringung nachzuladenden Programmcodes in Chipkarten sind z.B. im „Handbuch der Chipkarten" von W. Rankl, W. Effing, Hansa Verlag München, 3. Auflage, beschrieben. Die Programmerstellung erfolgt danach vollständig auf einem Hintergrundsystem. Der erstellte ausführbare Programmcode wird über eine, z.B. durch eine gegenseitige Authentisierung gesicherte Schnittstelle auf die Chipkarte übertragen. Aus Sicherheitsgründen erfolgt die Einbringung des ausführbaren Programmcodes auf die Chipkarte vorzugsweise online, nachdem zuvor eine eindeutige Identifizierung und Zuordnung von Hintergrundsystem, Schnittstellen, Kartenbetriebssystem und Kartenmikroprozessor erfolgt ist. Um bei Wahrung einer höchstmöglichen Sicherheit die Verwaltbarkeit der die Identifikationsinformationen enthaltenden Datenbanken der Hintergrundsysteme zu gewährleisten, werden die Genehmigungen zur Erstellung von ausführbaren Programmcode auf Hintergrundsystemen von den Kartenherausgebern nur unter Auflagen erteilt und werden die erteilten Genehmigungen gelistet. Die grundsätzlich geschaffene Möglichkeit, ausführbaren Programmcode für Chipkarten selbst zu erstellen, wird dadurch beschränkt.

Zur Sicherung eines über ein offenes Datennetz geführten Datenaustausches zwischen zwei Computern ist eine Anzahl von auf unterschiedlichen Verschlüsselungstechniken beruhenden Methoden bekannt, u.a. das SSL (Secure Socket Layer) Protokoll, PGP (Pretty Good Privacy), das Secure Messaging

oder das SMIME-Protokoll. Methoden dieser Art werden auch in dem nachfolgend beschriebenen, erfindungsgemäßen Verfahren genutzt, sind aber an sich nicht Gegenstand der Erfindung. Zu den Ausführungsdetails einschließlich der kryptologischen Realisierung wird deshalb allgemein auf die vielfältig verfügbaren Beschreibungen der jeweiligen Methoden in der einschlägigen Literatur sowie im Internet verwiesen. Dasselbe gilt für die üblichen im Zusammenhang mit Datensicherungsverfahren eingesetzten Mittel wie etwa die Verschlüsselung gemäß dem 3DES-Verfahren oder die Bildung von Message Authentication Codes (MAC).

Der Erfindung liegt die Aufgabe zugrunde, ein Verfahren anzugeben, das es bei Wahrung größtmöglicher Sicherheit gegen Datenmanipulation gestattet, einem möglichst großen Kreis von Nutzern die Erstellung von ausführbaren Programmen für programmierbare tragbare Datenträger zu erlauben. Aufgabe der Erfindung ist es weiterhin, die zur Ausführung des Verfahrens nötigen Systemkomponenten anzugeben.

Die Aufgabe wird gelöst durch ein Verfahren mit den Merkmalen des Hauptanspruchs. Erfindungsgemäß werden einem Nutzer ein Programmeditor zur Erstellung von Programmquelltexten sowie ein vorkomplettierter tragbarer Datenträger zur Verfügung gestellt, der über Softwarewerkzeuge zur Endbearbeitung verfügt, welche die Umwandlung von in einem Übergangsformat vorliegendem Transportcode in ausführbaren Programmcode erlauben. Die Erstellung eines ausführbaren Programmes für den Datenträger erfolgt verteilt. Mit dem Programmeditor erstellt der Nutzer einen Programmquelltext, der nachfolgend über eine gesicherte Verbindung an einen beim Herausgeber des Datenträgers befindlichen Computer übermittelt wird. Die gesicherte Verbindung kann dabei hergestellt werden, indem ein Programmquelltext von dem vorkomplettierten Datenträger selbst zu einem

Transportcode verschlüsselt und so gegen Veränderung gesichert wird, daß nur ein bestimmter, über einen bei einem Herausgeber des Datenträgers befindlichen Computer adressierter Empfänger den Transportcode entschlüsseln und auf Integrität überprüfen kann.

Aus dem eingegangenen Programmquelltext erzeugt der beim Herausgeber des Datenträgers befindliche Computer durch Kompilieren und Linken einen ausführbaren Programmcode. Bestandteil des Kompilier- und Linkvorganges ist eine formale Verifikation der erzeugten Programmcodes, durch die insbesondere aggressiver Code ermittelt wird. Den verifizierten, ausführbaren Programmcode setzt der beim Herausgeber des Datenträgers befindliche Computer in ein Übergangsformat um und übermittelt dieses über den Computer des Nutzers an den vorkomplettierten, tragbaren Datenträger. Dieser überführt ihn mit Hilfe der Endbearbeitungsoftwarewerkzeuge wieder in ausführbaren Programmcode und übernimmt diesen in seinen Speicher.

Vorzugsweise sind die sicherheitsrelevanten Teile der Endbearbeitungssoftware im vorkomplettierten Datenträger enthalten. Damit werden zweckmäßig im vorkomplettierten Datenträger selbst insbesondere die Entschlüsselung und/oder die Feststellung der Authentizität und/oder der Integrität eines einen Programmcode enthaltenden Transportcodes ausgeführt, bevor im fehlerfreien Fall der resultierende, ausführbare Programmcode in den Speicher des Datenträgers abgelegt wird.

Das erfindungsgemäße Verfahren schafft eine sichere „Ende-zu-Ende"-Verbindung zwischen einem bei einem Herausgeber befindlichen Computer und einem Datenträger über einen bei einem Nutzer befindlichen Computer. Durch die Gestaltung der Vorkomplettierung und die Wahl der Software-

werkzeuge läßt es sich dabei leicht an den Typ und die technischen Möglichkeiten der jeweils gegebenen Datenträger anpassen. Sind die Datenträger nur für die Ausführung von symmetrischen Verschlüsselungsverfahren eingerichtet, erfolgt die Herstellung einer gesicherten „Ende-zu-Ende"-Verbindung zweckmäßig durch Verwendung eines symmetrischen karteninidividuellen Schlüssels einerseits, und einer überlagerten, vereinfachten asymmetrischen Verschlüsselung auf der Datenverbindung zwischen dem Computer des Nutzers und dem beim Herausgeber des Datenträgers befindlichen Computer andererseits. In einer alternativen Ausführung erfolgt zur Sicherung der Datenübertragung zwischen dem Computer des Nutzers und dem beim Herausgeber des Datenträgers befindlichen Computer eine asymmetrische Verschlüsselung mit wechselseitiger Authentifizierung und wird die gesicherte „Ende-zu-Ende"-Verbindung zwischen dem beim Herausgeber befindlichen Computer und dem Datenträger mit den Mechanismen des Secure-Messaging eingerichtet.

Sind die Datenträger für die Ausführung von asymmetrischen Verschlüsselungsverfahren eingerichtet, wird zweckmäßig zwischen dem beim Herausgeber des Datenträgers befindlichen Computer und dem Datenträger direkt eine durch asymmetrische Verschlüsselung gesicherte „Ende-zu-Ende"-Verbindung ausgebildet. Der Computer des Nutzers fungiert dabei lediglich als Mittler.

Das erfindungsgemäße Verfahren hat den Vorteil, daß die Erstellung von ausführbaren Programmen für einen Datenträger grundsätzlich beliebigen Nutzern überlassen werden kann, ohne daß die Identität des Nutzers festgestellt und verwaltet werden müßte. Da der Herausgeber der Datenträger in jede Programmerstellung einbezogen wird, ist die Sicherheit der erzeugten Programme und darüber des gesamten Systems stets gewährleistet. Weil

insbesondere die Compilerfunktionalität beim Herausgeber der Datenträger verbleibt, muß wichtiges und sicherheitsrelevantes Know-How nicht an die Nutzer übergeben werden.

Durch Gestaltung der Compilerfunktionalität derart, daß direkte, unverschlüsselte Zugriffe auf von Nutzern stammende Programmquelltexte oder erzeugte ausführbare Programmcodes blockiert werden, läßt sich umgekehrt sicherstellen, daß anwendungsspezifisches Know-How der Nutzer vor dem Herausgeber geschützt wird. Zweckmäßig wird hierzu in dem beim Herausgeber befindlichen Computer ein Hardware-Sicherheitsmodul eingesetzt, in dem die Compilerfunktionalität, die Ver-/Entschlüsselung von Programmen, die Prüfung/Erstellung von Signaturen sowie die Authentisierung ausgeführt werden. Außerhalb des Hardware-Sicherheitsmodules erscheinen Programmquelltexte oder ausführbare Programmcodes nur in verschlüsselter Form.

Durch die formale Verifikation neu erstellter Programme beim Herausgeber, d.h. in sicherer Umgebung, kann ferner sehr zuverlässig die Einbringung aggressiver Programmcodes in mittels eines Datenträgers nutzbare Systeme verhindert werden. Zudem ergibt sich der Vorteil, daß alle erstellten ausführbaren Programme mit dem jeweils aktuellsten Compiler kompiliert werden. Das erfindungsgemäße Verfahren kann dabei online oder offline ausgeführt werden. Für die Herausgeber von Datenträgern eröffnet das erfindungsgemäße Verfahren sogar die Möglichkeit, die Erstellung der jeweils gewünschten lauffähigen Anwendungsprogramme gänzlich den Nutzern zu überlassen und die Datenträger überhaupt nur in vorkomplettierter Form auszuliefern. Die durch die verteilte Programmerstellung stets erzwungene Einbindung des Herausgebers in eine Programmerstellung ermöglicht desweiteren die Einführung von Nutzungsmethoden, die Gebührenmodelle

verwenden, welche z.B. auf der Zahl oder der Art der auf einen Datenträger gebrachten ausführbaren Programme beruhen.

Ein Ausführungsbeispiel der Erfindung wird nachfolgend unter Bezugnahme auf die Zeichnung näher erläutert.

Es zeigen:

Fig. 1 ein System zur Ausführung einer Programmerstellung,

Fig. 2 die Struktur des integrierten Schaltkreises eines programmierbaren, tragbaren Datenträgers,

Fig. 3 die Struktur eines zweiten Computers,

Fig. 4 den grundlegenden Ablauf einer verteilten Programmerstellung,

Fig. 5 bis 7 Flußdiagramme zur Veranschaulichung des Ablaufes einer Programmerstellung,

Fig. 8 das Prinzip einer Online-Prüfung eines erstellten Programmes auf Lauffähigkeit.

Fig. 1 veranschaulicht die grundlegende Struktur eines Systems zur verteilten Erstellung eines Programmes für einen programmierbaren, tragbaren Datenträger. Ein erster, für einen Datenaustausch mit einem tragbaren Datenträger 10 ausgebildeter Computer 20 ist über eine Datenverbindung 28 mit einem zweiten Computer 30 verbunden.

Der erste Computer 20 befindet sich bei einem Nutzer, etwa bei einer Bank, einer Versicherung, einem Einzelhändler, einer medizinischen Einrichtung oder dergleichen oder bei einem Dienstleister, der im Auftrag der vorgenannten Einrichtungen Programme erstellt. Er besitzt eine erste, kontaktierend oder berührungslos arbeitende Schnittstelle 24, die z.B. als Kontaktfeld, in Form einer Spule oder als optischer Signalgeber realisiert sein kann und die einen Datenaustausch mit einem tragbaren Datenträger 10 ermöglicht. Über eine weitere Schnittstelle 26 ist er an eine Datenverbindung 28 angeschlossen. Über beide Schnittstellen 24, 26 verbindet der Nutzercomputer 20 den Datenträger 10 mit der Datenverbindung 28. Der Nutzercomputer 20 stellt dem Datenträger 10 dabei Zusatzfunktionen bereit. Insbesondere gestattet er den Betrieb eines, im folgenden kurz Editor genannten, Editierungsprogrammes 22, das die Erstellung von Quelltexten von Programmen für einen Datenträger 10 erlaubt.

Für den programmierbaren, tragbaren Datenträger 10 wird anschließend die Form einer Chipkarte zugrundegelegt. Auf diese Erscheinungsform ist er aber keineswegs beschränkt. Der Datenträger 10 kann vielmehr, angepaßt an die jeweilige Nutzung, auch anders ausgebildet sein, etwa in Gestalt einer Uhr, als Schreibmittel usw. Unabhängig von seiner konkreten Erscheinungsform besitzt der tragbare Datenträger 10 jeweils eine zur Schnittstelle 24 des Nutzercomputers 20 korrespondierende Schnittstelle 14, welche einen Datenaustausch mit einem Nutzercomputer 20 ermöglicht. Desweiteren besitzt der tragbare Datenträger 10 einen integrierten Schaltkreis 12, welcher eine zentrale Prozessoreinheit sowie einen Speicher zur Aufnahme des Programmcodes wenigstens eines durch die zentrale Prozessoreinheit ausführbaren Anwendungsprogrammes aufweist.

Der zweite Computer 30 befindet sich typischerweise bei einem Herausgeber von tragbaren Datenträgern 10 oder bei einem autorisierten Betreiber des hier beschriebenen Verfahrens. In der Regel besitzt er eine im Vergleich zu der des Nutzercomputer 20 bzw. des tragbaren Datenträgers 10 wesentlich größere Rechenleistung. Der zweite Computer 30 muß dabei nicht als bauliche Einheit realisiert sein. Er kann vielmehr auch als System mit verteilten Komponenten ausgeführt sein, welche über ein spezielles Datennetz verbunden sind. Zur Speicherung bzw. zur Ausführung sicherheitskritischer Funktionen können Hardware-Sicherheitsmodule eingesetzt sein. Über eine Schnittstelle 34 ist der zweite Computer 30 an die Datenverbindung 28 angeschlossen. Der zweite Computer 30 ist insbesondere dazu ausgebildet, ein Kompilierungsprogramm 310 zur Umsetzung eines in einer Programmierhochsprache vorliegenden Quelltextprogrammes in Maschinensprache auszuführen; er wird deshalb nachfolgend als Compilerserver bezeichnet.

Die Datenverbindung 28 hat üblicherweise die Gestalt eines Datennetzes und kann insbesondere durch das Internet realisiert sein. Obwohl in Fig. 1 nur eine Verbindung zwischen zwei Komponenten 20, 30 gezeigt ist, können über die im folgenden Datennetz genannte Datenverbindung 28 auch mehrere Nutzercomputer 20 mit einem oder auch mehreren Compilerservern 30 verbunden sein.

Fig. 2 zeigt die Struktur des integrierten Schaltkreises 12 einer Chipkarte 10 mit als Vorkomplettierung aufgebrachten Softwarewerkzeugen. Der integrierte Schaltkreis 12 besitzt eine für Chipkartenprozessoren typische Architektur und weist eine zentrale Prozessoreinheit 100, einen flüchtigen Arbeitsspeicher 102, sowie eine nichtflüchtige Speicheranordnung 104 auf, letztere bestehend aus einem nichtflüchtigen Nur-Lese-Speicher sowie einem nichtflüchtigen, wiederbeschreibbaren Speicher. Üblicherweise ist der flüchtige

Arbeitsspeicher 102 ein RAM-Speicher, der nichtflüchtige Nur-Lese-Speicher ein ROM-Speicher und der nichtflüchtige, überschreibbare Speicher ein EEPROM-Speicher. Außer diesen genannten können beliebige andere, dieselbe Funktionalität aufweisenden Speichertypen eingesetzt werden. Die zentrale Prozessoreinheit 100 ist ferner mit der Schnittstelle 14 verbunden.

In der nichtflüchtigen Speicheranordnung 104 befindet sich eine Anzahl von zur Nutzung des Datenträgers 10 benötigten Softwarewerkzeugen, welche in einer Vorkomplettierungsphase vor Übergabe des Datenträgers 10 an einen Nutzer angelegt werden. Unter Softwarewerkzeugen sollen hierbei alle nicht durch einen Nutzer veränderbaren Programme, Routinen oder Datensätze verstanden werden, die bedarfsweise zur Ausführung jeweils bestimmter Datenverarbeitungsaufgaben einsetzbar sind. Im Rahmen der Vorkomplettierung angelegt wird dabei zum einen eine ausführungsunabhängige, stets gleichartige Kartengrundausstattung 110. Sie umfaßt zumindest das Betriebssystem 111, einen Basisprogrammcode 112 zur Realisierung von Anwendungen, die sich bereits bei Übergabe an den Nutzer auf der Chipkarte 10 befinden, sowie einen Speicherbereich 113 zur späteren Aufnahme von nachgeladenem, ausführbarem Programmcode.

Zum anderen wird eine auf die jeweils gewählte Ausführungsvariante abgestimmte Auswahl der folgenden Softwarewerkzeuge angelegt: eine für den integrierten Schaltkreis 12, und damit für die Chipkarte 10, individuelle und eindeutige Identifikationsinformation 114, z.B. eine Seriennummer, ein Programm 116 zur Ausführung asymmetrischer kryptographischer Algorithmen, ein Programm 118 zur Durchführung symmetrischer kryptographischer Algorithmen, ein Programm 120 zur Führung eines Datenaustausches nach dem Prinzip des Secure Messagings, ein Programm 122 zur Durchführung eines Datenaustausches über das Datennetz 28 gemäß dem SSL-

Protokoll, ein chipkartenindividueller Signaturschlüssel 124, ein chipkartenindividueller symmetrischer Schlüssel 126, der öffentliche Schlüssel 128 eines der Chipkarte 10 zugeordneten Compilerservers 30, ein privater Kartenschlüssel 130 zur Verwendung in einem asymmetrischen Verschlüsselungsverfahren, ein Zertifikat 132, welches die Zusammengehörigkeit zwischen öffentlichen Kartenschlüsseln und Identifikationsinformationen mit einer Signatur eines Herausgebers bestätigt, Speicherraum zur Aufnahme eines Sitzungssschlüssel 134 - dieser wird im Unterschied zu den vorgenannten Schlüsseln bei der Aufnahme eines Datenaustausches mit einem Compilerserver 30 jeweils neu erzeugt, sowie ein Sequenzzähler 136. Alle genannten Softwarewerkzeuge können jeweils auch mehrfach vorhanden sein. Das gilt besonders für die aufgeführten Schlüssel, Zertifikate und den Sequenzzähler.

Fig. 3 veranschaulicht die Struktur eines Compilerservers 30 mit den bei der Durchführung einer Programmerstellung eingesetzten Programmen und Softwarewerkzeuge. Kern des Compilerservers 30 bildet eine zentrale Prozessoreinheit 300, welche über eine Schnittstelle 34 mit dem Datennetz 28 verbunden ist, um darüber einen Datenaustausch mit einem Nutzercomputer 20 und darüber mit einer Chipkarte 10 zu führen. Weiter sind der zentralen Prozessoreinheit 300 ein flüchtiger Arbeitsspeicher 302, in der Regel in Gestalt eines RAM-Speichers, sowie eine nichtflüchtige Speicheranordnung 304 zugeordnet, welche üblicherweise einen Nur-Lese-ROM-Speicher sowie einen Massenspeicher, etwa eine Festplatte, umfaßt.

In der Speicheranordnung 304 sind die zur Durchführung des vorgeschlagenen Verfahrens benötigten Softwarewerkzeuge abgelegt. Fig. 3 zeigt der Einfachheit wegen eine Übersicht über sämtliche im Zusammenhang mit dieser Beschreibung in Betracht kommenden Softwarewerkzeuge. Die Auswahl der

tatsächlich benötigten Softwarewerkzeuge hängt, wie bei der Chipkarte 10, von der zur Realisierung des Verfahrens konkret gewählten Ausführungsform. Allgemein können sich in der Speicheranordnung 304 an Softwarewerkzeugen befinden: Ein, in Fig. 3 Compiler genanntes Compilierungsprogramm 310 zur Umsetzung von Programmquelltext in einen Programmcode, ein, in Fig. 3 Linker genanntes Linkprogramm 312 zur Einbindung von bereits erstellten Programmcodes in den Kontext eines neu erstellten Programmes, eine Codebibliothek 318 mit dem Programmcode bereits vorhandener Programme und Programmteile, eine Datenbank 320 zur Ablage von bestimmten Nutzern zugeordneten Programmcodes, ein Debug-Programm 316 zur Prüfung eines erstellten Programmes auf Lauffähigkeit, ein Programm 321 zur formalen Verifikation von erzeugten Programmen und/oder Quelltexten, ein oder mehrere Hauptschlüssel 324, welche zu dem oder den chipkartenindividuellen, symmetrischen Schlüsseln 126 korrespondieren, ein oder mehrere Hauptschlüssel 326, welche zu den chipkartenindividuellen Schlüsseln 124 zur Bildung von Datensicherungscodes, insbesondere MACs korrespondieren, einen oder mehrere öffentliche Serverschlüssel 328 zur Durchführung von asymmetrischen kryptographischen Algorithmen, einen oder mehrere korrespondierende private Serverschlüssel 330, einen oder mehrere öffentliche Kartenschlüssel 332 zur Durchführung asymmetrischer Algorithmen, ein oder mehrere Serverzertifikate 334, ein oder mehrere Sequenzzähler 338, sowie eine Liste mit Zertifikaten, die bei der Herstellung der vorkomplettierten Chipkarte 10 gebildet wurden und in der Chipkarte 10 im Bereich 132 gespeichert werden. Desweiteren beinhaltet die Speicheranordnung 304 eine Nutzerliste 340 mit Identifikationsinformationen, die eine eindeutige Identifizierung einer Chipkarte ermöglichen; Identifikationsinformationen zur Identifizierung von Chipkarten 10 können beispielsweise deren Seriennummern sein.

Zweckmäßig werden in einem Compilerserver 30 bei der praktischen Umsetzung des Verfahrens von den vorgenannten Softwarewerkzeugen nur die tatsächlich benötigten eingerichtet, die jeweils nicht benötigten weggelassen.

Bedeutung und Verwendung der in der vorkomplettierten Chipkarte 10 bzw. dem Compilerserver 30 vorhandenen Softwarewerkzeuge werden nachfolgend anhand der Fig. 4, die den grundlegenden Ablauf einer verteilten Programmerstellung zeigt, sowie der Fig. 5 bis 7 erläutert, die drei Ausführungsformen einer verteilten Programmerstellung veranschaulichen.

Fig. 4 zeigt zunächst den grundlegenden Ablauf einer verteilten Programmerstellung. In einer Vorbereitungsphase werden einem Nutzer eine durch Aufbringen von Softwarewerkzeugen vorkomplettierte Chipkarte 10 sowie ein Editor 22 zur Verfügung gestellt, Schritt 400. Mit dem Editor 22 erstellt er auf dem Nutzercomputer 20 einen Programmquelltext Q, Schritt 402. Durch Anwendung einer geeigneten Verschlüsselungstechnik wird dieser mit einer Transportsicherung versehen, Schritt 404, und in einen Transportcode T, TQ, TQ_{SSL} überführt, Schritt 406. Der Transportcode T, TQ, TQ_{SSL} wird an den Compilerserver 30 übermittelt; Schritt 408.

Der Compilerserver 30 hebt durch Entschlüsselung die Transportsicherung auf, Schritt 410, und gewinnt den in dem Transportcode T, TQ, TQ_{SSL} enthaltenen Programmquelltext Q zurück, Schritt 412. Den Programmquelltext Q kompiliert, bindet und verifiziert er anschließend, Schritt 414. Es resultiert ein lauffähiger Programmcode C, Schritt 416, der nachfolgend wiederum transportgesichert wird, Schritt 418. Er wird hierzu durch Anwendung geeigneter Verschlüsselungsmechanismen, die nicht mit den zuvor auf Seiten des Nutzercomputers 20 angewandten übereinstimmen müssen, in ein Übergangsformat U, U_{SM}, U_{SSL} überführt, Schritt 420. In diesem Übergangs-

format wird er über den Nutzercomputer 20 an die Chipkarte 10 übermittelt, Schritt 422.

Jene ermittelt unter Verwendung der bei der Vorkomplettierung angelegten Softwarewerkzeuge aus dem im Nutzercomputer 20 eingegangenen Transportcode U, U_{SM}, U_{SSL} durch Entschlüsselung wieder den lauffähigen Programmcode C und lädt diesen schließlich in seinen Speicher.

Fig. 5 zeigt eine verteilte Programmerstellung, bei der die Datensicherheit durch Nutzung von auf der Chipkarte 10 vorbereiteten Mitteln in Wechselwirkung mit dem Compilerserver 30 erreicht wird. Die in Fig. 5 dargestellte Ausführungsform eignet sich besonders für Systeme, in denen die verwendeten Chipkarten 10 nur symmetrische Verschlüsselungstechniken beherrschen.

Fig. 6 zeigt eine Ausführungsform, bei der die zwischen Nutzercomputer 20 und Compilerserver 30 über das Datennetz 28 erfolgende Datenübertragung mittels eines SSL-Protokolls gesichert ist, während der direkt zwischen Chipkarte 10 und Compilerserver 30 erfolgende Datentransport gemäß dem Secure-Messaging Mechanismus ausgeführt wird. Die Ausführungsform eignet sich ebenfalls für Systeme, in denen die verwendeten Chipkarten 10 nur symmetrische Verschlüsselungstechniken erlauben.

Fig. 7 veranschaulicht eine Ausführungsform, bei der der Nutzercomputer 20 im wesentlichen nur als Mittler zwischen Chipkarte 10 und Compilerserver 30 wirkt. Die Sicherung der zwischen Chipkarte 10 und Compilerserver 30 transportierten Daten erfolgt, indem zwischen Compilerserver 30 und Chipkarte 10 unter Verwendung des SSL-Protokolls direkt eine gesicherte „Ende-zu-Ende"-Verbindung eingerichtet wird.

Tabelle 1 veranschaulicht systematisch die Anwendbarkeit der drei nachfolgend anhand der Fig. 5, 6, 7 beschriebenen Ausführungsformen in Abhängigkeit von der Durchführung der Datenübertragung, der Ausstattungsanforderungen an die Chipkarte 10 und der Art der Transportsicherung.

Tabelle 1

Verfahren	Datenübertragung	Anforderung an Chipkarte	Art der Transportsicherung
Fig 5	Offline	Nur symmetrische Algorithmen	Verschlüsselung und MAC durch Chipk.
Fig. 6	Online	Symmetrische und/oder asymmetrische Algorithmen	Secure Messaging durch Chipkarte
Fig. 7	Online	Symmetrische und asymmetrische Algorithmen	SSL durch Chipkarte

Die linke Spalte in den Fig. 5, 6, 7 zeigt jeweils die Aktivitäten des Compilerservers 30, die rechte die Aktivitäten des Nutzercomputers 20 bzw. der Chipkarte 10, wobei „N" den Nutzercomputer 20 bezeichnet, „K" die Chipkarte 10.

Der in Fig.5 dargestellten Programmerstellung vorgeschaltet ist eine Vorbereitungsphase. Darin werden dem Nutzer durch den Herausgeber eine vorkomplettierte Chipkarte 10, Schritt 500, sowie ein Editor 22 zur Einrichtung auf seinem Computer 20, Schritt 502, bereitgestellt. Auf der vorkomplettierten Chipkarte 10 befinden sich bzw. sind neben der Grundausstattung 113 eingerichtet: eine Identifikationsinformation ID im Speicherbereich 114, ein

Programm 118 zur Durchführung von symmetrischen Kryptoalgorithmen, etwa des „3DES"-Algorithmus, mindestens ein kartenindividueller Schlüssel K_{MAC} zur Bildung eines Datensicherungscodes, vorzugsweise in Gestalt eines MACs, im Speicherbereich 124, mindestens ein Schlüssel K_{ENC} zur symmetrischen Verschlüsselung im Speicherbereich 126, sowie Speicherplatz 134 zur Aufnahme wenigstens zweier Sitzungsschlüssel SK_{ENC}, SK_{MAC}. Weiter sind auf der vorkomplettierten Chipkarte 10 wenigstens zwei Sequenzzähler 136 mit Werten SEQ_C, SEQ_H eingerichtet. Der Sequenzzähler SEQ_C dient dabei zur Berechnung der Sitzungsschlüssel SK_{ENC}, SK_{MAC} die zur sicheren Übertragung von Programmquelltexten Q vom Nutzercomputer 20 zum Compilerserver 30 genutzt werden, der Sequenzzähler SEQ_H dient zur Berechnung von Sitzungsschlüsseln SK_{ENC}, SK_{MAC}, die zur sicheren Übertragung von Programmcodes C vom Compilerserver 30 zum Nutzercomputer 20 genutzt werden.

Auf dem Compilerserver 30 werden für jede ausgegebene Chipkarte 10 zwei Sequenzzähler 338 mit Werten den SEQ_C, SEQ_H eingerichtet. Die Werte SEQ_C, SEQ_H stimmen, damit der Compilerserver 30 nach der gleichen Rechenvorschrift, wie sie die Chipkarte 10 einsetzt, die Sitzungsschlüssel SK_{ENC}, SK_{MAC} berechnen kann, stets mit den Werten der korrespondierenden Sequenzzähler 136 der zugeordneten Chipkarte 10 überein. Zur Erhöhung der Sicherheit werden bei jeder Übertragung von Programmquelltexten Q bzw. Programmcodes C andere Sitzungsschlüssel SK_{ENC}, SK_{MAC} verwendet. Zu diesem Zweck erhöhen Chipkarte 10 und Compilerserver 30 jeweils vor der Berechnung von Sitzungssschlüssel SK_{ENC}, SK_{MAC} die Werte SEQ_C bzw. SEQ_H der Sequenzzähler 136, 338.

Der Editor 22 erlaubt die Erstellung von Programmquelltext Q, z.B. in einer Programmierhochsprache. Vorzugsweise unterstützt er die Programmer-

stellung durch eine graphisch unterlegte, dialoggesteuerte Eingabeführung und bietet direkt nutzbare Entwicklungshilfsmittel wie eine Syntaxprüfung oder die Einbindung von Programmschnittstellen zur Codebibliothek.

Stehen Chipkarte 10 in vorkomplettierter Form und Nutzercomputer 20 bereit, erstellt der Nutzer unter Verwendung des Editors 22 den Programmquelltext Q eines zur Einbringung in eine Chipkarte 10 bestimmten Programmes, Schritt 504. Vorzugsweise erfolgt die Erstellung in einer Programmierhochsprache, generell ist aber auch jedes andere Format möglich. Ist ein Programmquelltext Q erstellt, beauftragt der Nutzer die Chipkarte 10 über den Editor 22 mittels eines entsprechenden Befehles, den Programmquelltext Q zu verschlüsseln und mit einem MAC gegen Veränderung zu sichern. Dazu erhöht die Chipkarte 10 zunächst den Sequenzzählerwert SEQ_C und generiert die Sitzungssschlüssel SK_{ENC} und SK_{MAC}, z.B. mit dem symmetrischen 3DES -Algorithmus, Schritt 506. Anschließend verschlüsselt sie den von dem Editor 22 über die Schnittstellen 24, 14 erhaltenen Programmquelltext Q mit dem Sitzungsschlüssel SK_{ENC} zu einem Zwischencode Q' und berechnet mit dem Sitzungsschlüssel SK_{MAC} einen MAC über Q', Schritt 508. Zwischencode Q' und MAC übergibt die Chipkarte 10 sodann über die Schnittstellen 24,14 zurück an den Editor 22.

Dieser ermittelt desweiteren die im Speicherbereich 114 der Chipkarte 10 angelegte Kartenidentifikation ID, Schritt 509, und fügt sie mit dem Zwischencode Q` sowie dem MAC zu einem Transportcode T zusammen. Den derart gebildeten Transportcode T übermittelt der Nutzercomputer 20 über das Datennetz 28 an den Compilerserver 30, Schritt 510. Die Übertragung des Transportcodes T kann dabei in beliebiger Weise über ein ungesichertes Medium erfolgen. Beispielsweise kann der Transportcode T als E-mail übertragen oder auf Diskette per Post an den Herausgeber geschickt werden. Da-

neben kann der Transportcode T auch online über das Datennetz 28 an den Compilerserver 30 geschickt werden. Vertraulichkeit und Integrität des übermittelten Transportcodes T werden durch die Verschlüsselung und die MAC Berechnung durch die Chipkarte 10 gewährleistet.

Beim Compilerserver 30 eingegangen, prüft dieser zunächst, Schritt 512, ob die im Transportcode T enthaltene Identifikationsinformation ID auch in der im Compilerserver 30 geführten Identifikationsliste 340 enthalten ist, die vorzugsweise eine Kundenliste bildet. Trifft das zu, leitet er zunächst aus den in den Speicherbereichen 324, 326 befindlichen Masterschlüsseln MK_{ENC} und MK_{MAC} mit Hilfe der Identifikationsinformation ID die zugehörigen kartenindividuellen Schlüssel K_{ENC} und K_{MAC} ab, Schritt 514. Aus diesen Schlüsseln sowie dem inkrementierten Sequenzzähler SEQ_C berechnet der Compilerserver 30 sodann die Sitzungsschlüssel SK_{ENC} und SK_{MAC} nach derselben Rechenvorschrift, die zuvor die Chipkarte 10 verwendet hat. Mit dem Sitzungsschlüssel SK_{MAC} berechnet der Compilerserver 30 anschließend seinerseits einen MAC`, Schritt 516, und vergleicht ihn mit dem in dem Transportcode T enthaltenen MAC. Stimmen beide überein, erkennt der Compilerserver 30 den Transportcode T als authentisch, d.h. von der Chipkarte 10 mit der Identifikationsinformation ID kommend, und integer, d.h. nicht bei der Übertragung verändert.

Hat der Compilerserver 30 einen Transportcode T als authentisch erkannt, entschlüsselt er den im Transportcode T enthaltenen Programmquelltext Q' mittels des Sitzungsschlüssels SK_{ENC}. Aufgrund der zuvor festgestellten Integrität des Transportcodes T stimmt das resultierende, entschlüsselte Format mit dem auf dem Nutzercomputer 20 ursprünglich erstellten Programmquelltext Q überein.

Den wiederhergestellten Programmquelltext Q überführt der Compilerserver 30 unter Verwendung des Kompilierungsprogrammes 310 in ein Zwischenformat, das er anschließend mittels des Linkprogrammes 312 unter Zugriff auf die Codebibliothek 318 mit bereits vorhandenem Programmcode verbindet, Schritt 518.

Kompilierungsprogramm 310 und Linkprogramm 312 sind in einer zweckmäßigen Gestaltung in Form eines Hardware-Sicherheitsmodules ausgeführt, welches die Compilier- und Linkfunktionalität, die Ent- und Verschlüsselung der bearbeiteten Programmdaten, die Prüfung und Erstellung von Signaturen sowie die Authentisierung beinhaltet. Alle bearbeiteten Programmdaten, insbesondere eingehende Programmquelltexte Q und erzeugte ausführbare Programmcodes C erscheinen dann außerhalb des Hardware-Sicherheitsmodules nur in verschlüsselter Form. Auf diese läßt sich sicherstellen, daß anwendungsspezifisches Know-How der Nutzer gegen Einsicht und Zugriffe über den Compilerserver 30 geschützt wird.

Zweckmäßig kann im Compilerserver 30 desweiteren eine Beschränkung des Zugriffes auf die Codebibliothek 318 eingerichtet sein, welche z.B. die Einbindung schon vorhandenen Programmcodes in einen neu erzeugten durch das Linkprogramm 312 beschränkt.

Der nach Compilierung und Linken resultierende Programmcode C wird mittels des Verifikationsprogrammes 321 formal verifiziert. Dabei wird der Programmcode C auf offensichtliche Fehler geprüft, etwa auf Einhaltung des Adreßraumes, auf Beachtung der vorgegebenen Speichergrößen, auf Typverletzungen oder auf Aggressivität, Schritt 520.

Ist der aus dem Programmquelltext Q erzeugte Programmcode C danach verwendungsfähig, d.h. durch die Chipkarte 10 ausführbar, wird er für die Rückübertragung in einen Transportcode U umgewandelt. Hierzu wird zunächst der Sequenzzählerwert SEQ_H erhöht. Mit dem erhöhten Sequenzzählerwert SEQ_H werden anschließend aus den Masterschlüsseln MK_{ENC} bzw. MK_{MAC} und der Identifikationsinformation ID die kartenindividuellen Schlüssel K_{ENC} und K_{MAC} abgeleitet und damit wiederum Sitzungsschlüssel SK_{ENC} und SK_{MAC} berechnet, Schritt 522. Die Berechnung der Sitzungsschlüssel SK_{ENC}, SK_{MAC} durch den Compilerserver 30 erfolgt auf dieselbe Weise wie sie zuvor, in Schritt 506, vom Nutzercomputer 20 vorgenommen wurde, wobei lediglich anstelle des Sequenzzählerwertes SEQ_C der Sequenzzählerwert SEQ_H verwendet wird.

Nachfolgend wird mit dem Sitzungsschlüssel SK_{ENC} der Programmcode C zu einem Zwischencode C' verschlüsselt und über den Zwischencode C' mittels des Sitzungsschlüssel SK_{MAC} weiterhin ein MAC`` berechnet, Schritt 524. Zwischencode C' und MAC`` werden sodann zu einem Transportcode U zusammengefügt, den der Compilerserver 30 an den Nutzercomputer 20 übersendet, Schritt 526. Für die Übersendung des Transportcodes U kann wie im Falle des Transportcodes T ein beliebiges, insbesondere auch ein an sich unsicheres Übertragungsmedium wie eine Diskette oder der Versand per E-mail gewählt werden. Selbstverständlich ist daneben auch die Nutzung einer Online-Verbindung über ein Datennetz 28 möglich. Wird eine Online-Verbindung genutzt, besteht die Möglichkeit, zu einem Auftrag, d.h. der Absendung eines in einem Transportcode enthaltenen Programmquelltextes Q an einen Compiler 30, in einer einzelnen Online-Sitzung auch das Ergebnis, d.h. einen Transportcode U mit dem Programmcode C` zu erhalten.

Der Nutzercomputer 20 leitet den erhaltenen Transportcode U über die Schnittstellen 24, 14 weiter an die Chipkarte 10, Schritt 528. Jene erhöht den Wert SEQ_H des Sequenzzählers 136, erzeugt damit auf dieselbe Weise wie zuvor der Compilerserver 30 in Schritt 522 die Sitzungssschlüssel SK_{ENC} bzw. SK_{MAC} und prüft, ob der mit Transportcode U übermittelte MAC`` identisch dem MAC ist, den die Chipkarte 10 selbst mittels des Schlüssels $SK_{MAC}$ aus U berechnen kann, Schritt 530. Stimmen MAC`` und MAC überein, ist der MAC`` aus U erfolgreich verifiziert. Da außer der Chipkarte 10 selbst nur der Compilerserver 30 über die Möglichkeit verfügt, den Schlüssel SK_{MAC} zu benutzen, ist der durch Entschlüsselung des in dem Transportcode U übermittelten Programmcodes C' gewonnenen Programmcode C authentisch, d.h. er wurde vom Compilerserver 30 aus einen von derselben Chipkarte 10 transportgesicherten Programmquelltext Q generiert. Der als authentisch erkannte Programmcode C wird von der Chipkarte 10 in den Kartenspeicher 113 geladen, Schritt 532.

Fig. 6 zeigt als Flußdiagramm eine Ausführungsform einer verteilten Programmerstellung, bei der die zwischen Nutzercomputer 20 und Compilerserver 30 über das Datennetz 28 erfolgende Datenübertragung mittels SSL gesichert ist, während der direkte Datentransport zwischen Chipkarte 10 und Compilerserver 30 mit Hilfe des Secure-Messaging-Mechanismus ausgeführt wird. Die Ausführungsform eignet sich wie die in Fig. 5 dargestellte Ausführungsform besonders für eine Online-Ausführung in Systemen, in denen die verwendeten Chipkarten 10 nur symmetrische Verschlüsselungstechniken erlauben.

Eine zur Durchführung der zweiten Ausführungsform vorkomplettierte Chipkarte 10 umfaßt neben der Grundausstattung 110 mit Betriebssystem 111, Basisprogrammcode 112 und Speicherraum 113 für Komplettierungs-

programmcode, eine Routine 120 zur Durchführung des Secure-Messagings, einen privaten Kartenschlüssel 130 sowie einen öffentlichen Serverschlüssel 128. Der Nutzercomputer 20 verfügt ferner über die Programmfunktionalität zur Ausführung des SSL-Protokolls ohne Authentisierung von Chipkarten.

Die Durchführung einer Programmerstellung in der zweiten Ausführungsform entspricht zunächst der ersten Ausführungsform gemäß Fig. 5 und umfaßt die Schritte 500 bis 504.

Liegt ein Programmquelltext Q vor, richtet der Nutzer über das Datennetz 28 eine Verbindung zwischen seinem Computer 20 und dem Compilerserver 30 des Herausgebers ein, Schritt 600.

Ist die physikalische Verbindung zum Compilerserver 30 hergestellt, wird zwischen Nutzercomputer 20 und Compilerserver 30 ein SSL-Protokoll gestartet. Nutzercomputer 20 und Compilerserver 30 bestimmen dabei jeweils einen Sitzungsschlüssel SK_{SSL}, Schritte 601, 602. Anschließend wird innerhalb des SSL-Protokolls ein sogenannter IP-Tunnel zwischen Compilerserver 30 und Chipkarte 10 zur Ausführung des Secure-Messagings eingerichtet, Schritt 604. Im Nutzercomputer 20 wird dabei das von der Chipkarte 10 durchgeführte Protokoll des Secure-Messagings in das, nur zwischen Nutzercomputer 20 und Compilerserver 30 eingesetzte SSL-Protokoll eingebettet. In dem IP-Tunnel werden nachfolgend chipkartenspezifische Datensätze, vorzugsweise in Gestalt von APDUs (Application Protocol Data Unit) direkt zwischen Chipkarte 10 und Compilerserver 30 transportiert. In Bezug auf das Secure-Messaging fungiert der Nutzercomputer 20 nur als reiner Mittler.

Gemäß dem Secure-Messaging führen Chipkarte 10 und Compilerserver 30 sodann eine wechselseitige Authentifizierung durch, wobei sich zunächst die

Karte 10 gegenüber dem Compilerserver 30 authentisiert, Schritt 606, anschließend der Compilerserver 30 gegenüber der Karte 10, Schritt 608. Verläuft die wechselseitige Authentifizierung zwischen Chipkarte 10 und Compilerserver 30 erfolgreich, wird die Nutzung aller Funktionen des Compilerservers 30 mittels des Nutzercomputers 20 freigegeben, Schritt 610.

Liegt die Nutzungsfreigabe vor, verschlüsselt der Nutzercomputer 20 den erstellten Programmquelltext Q mit dem vorher bestimmten Sitzungsschlüssel SK_{SES} und übermittelt den daraus resultierenden Transportcode TQ an den Compilerserver 30, Schritt 612.

Im Compilerserver 30 eingegangen wird der Transportcode TQ mit Hilfe des zuvor im Compilerserver 30 generierten Sitzungsschlüssel SK_{SSL} wieder entschlüsselt, Schritt 614, und in den auf dem Nutzercomputer 20 erstellten Quelltext Q überführt. Zweckmäßig erfolgt die Ausführung der Schritte 610, 612, 614 in Form eines kontinuierlichen Datenaustausches zwischen Nutzercomputer 20 und Compilerserver 30, so daß die Wiederherstellung des Quelltextes Q im Compilerserver 30 unmittelbar nach Erhalt des letzten verschlüsselten Quelltextdatensatzes vom Nutzercomputer 20 abgeschlossen wird.

Aus dem Quelltext Q erzeugt der Compilerserver 30 sodann durch Ausführung der anhand der Fig. 5 beschriebenen Schritte 518 und 520 einen ausführbaren Programmcode C.

Den ausführbaren Programmcode C wandelt der Compilerserver 30 durch Anwendung der Secure-Messaging-Mechanismen in gesicherten Programmcode C_{SM}, Schritt 620. Den gesicherten Programmcode C_{SM} überführt er anschließend durch Verschlüsselung mit Hilfe des Sitzungsschlüssel SK_{SES}

in einen in einem Übergangsformat vorliegenden Transportcode UC_{SM}, Schritt 622. Durch die Verschlüsselung mit dem Sitzungsschlüssel SK_{SES} wird der, typischerweise in Gestalt von APDUs vorliegende, gesicherte Programmcode C_{SM} in eine Sicherung der Datenübertragung über das Datennetz 28 zwischen Compilerserver 30 und Nutzercomputer 20 eingebettet.

Den im Übergangsformat vorliegenden Transportcode UC_{SM} übermittelt der Compilerserver 30 an den Nutzercomputer 20. Dieser entschlüsselt UC_{SM} mittels des Sitzungssschlüssels SK_{SES}, Schritt 626, wodurch die zum Schutz der Datenübertragung zwischen Compilerserver 30 und Nutzercomputer 20 angebrachte Sicherung wieder entfernt wird. Den danach vorliegenden entschlüsselten, gemäß dem Secure-Messaging gesicherten Programmcode C_{SM} übergibt der Nutzercomputer 20 an die Chipkarte 10, Schritt 624.

In der Chipkarte 10 wird der gesicherte Programmcode C_{SM} durch Anwendung der umkehrenden Secure-Messaging-Mechanismen wieder in ausführbaren Programmcode C zurückgeführt, Schritt 628, und schließlich in die Speicheranordnung 104 in den dort zur Aufnahme von Komplettierungsprogrammcode vorbereiteten Bereich 113 geladen, Schritt 630.

Aus Gründen der Klarheit wurde vorstehend der in Fig. 6 dargestellte Verfahrensablauf als sequentielle Folge von separaten Schritten beschrieben. In der Praxis beinhalten die zwischen Chipkarte 10, Nutzercomputer 20 und Compilerserver 30 erfolgenden Datenübertragungen in der Regel einen Datenaustausch in jeweils beiden Richtungen. Sinnvoll ist es zudem, Verfahrensschritte, für die das möglich ist, in Form eines kontinuierlichen, quasiparallelen Datenaustausch- und Verarbeitungsprozesses auszuführen, in dem Compilerserver 30 und Nutzercomputer 20 bzw. Chipkarte 10 Verfahrensschritte zeitlich überlagernd ausführen. Zweckmäßig ist dies z.B. für die

Schritte 620 bis 630: sie werden vorzugsweise in Gestalt eines kontinuierlichen Datenaustausches zwischen Compilerserver 30 und Nutzercomputer 20 ausgeführt, in dem eine Übertragung von Datensätzen des Transportcodes UC_{SM} zum Nutzercomputer 20 bereits stattfindet, während auf dem Compilerserver 30 noch die Umsetzung des Programmcodes C gemäß dem Secure-Messaging erfolgt, und in dem die vom Compilerserver 30 über den Nutzercomputer 20 an die Chipkarte 10 übertragenen Datensätze durch diese unmittelbar vor dem Laden in den Speicherraum 113, d.h. ohne Zwischenspeicherung bis zum vollständigen Eingang, entschlüsselt werden.

Fig. 7 veranschaulicht eine weitere Ausführungsform der anhand der Fig. 4 beschriebenen Programmerstellung, bei der der Nutzercomputer 20 im wesentlichen nur als Mittler zwischen Chipkarte 10 und Compilerserver 30 wirkt. Die Sicherung der zwischen Chipkarte 10 und Compilerserver 30 transportierten Daten erfolgt, indem zwischen Compilerserver 30 und Chipkarte 10 unter Verwendung des SSL-Protokolls eine gesicherte, direkte „Ende-zu-Ende"-Verbindung eingerichtet wird.

Die Vorkomplettierung einer zur Durchführung dieser Ausführungsvariante geeigneten Chipkarte 10 beinhaltet neben der Einrichtung der Grundausstattung 110 mit Betriebssystem 111, Basisprogrammcode 112 und Speicherbereich für Komplettierungsprogrammcode 113 das Anlegen eines Programmes 122 zur Ausführung des SSL-Protokolls, die Hinterlegung eines Zertifikates 132, die Hinterlegung des privaten Kartenschlüssels 130 sowie die Hinterlegung des öffentlichen Serverschlüssels 128.

Die Durchführung des Verfahrens gemäß Fig. 7 entspricht zunächst der anhand der Fig. 5 beschriebenen Ausführungsform und umfaßt die Schritte 500 bis 504. Sie werden gefolgt von der Einrichtung einer Verbindung zwischen

der Chipkarte 10 und einem Compilerserver 30 über den Nutzercomputer 20, Schritt 700.

Chipkarte 10 und Compilerserver 30 führen nun ein vollständiges SSL-Protokoll aus. Innerhalb der Handshake-Prozedur erfolgt hierbei eine wechselseitige Authentifizierung, indem zum einen das Compilerserverzertifikat 332 durch die Chipkarte 10 geprüft wird, Schritt 701, zum anderen das in der Chipkarte 10 angelegte Zertifikat 132 durch den Compilerserver 30, Schritt 702. Ist nach der wechselseitigen Zertifikatsprüfung eine Weiterführung des Datenaustausches möglich, generieren Chipkarte 10 und Compilerserver 30 jeweils einen Sitzungsschlüssel, Schritt 704 bzw. 706.

Die in Fig. 7 veranschaulichte Ausführungsform eignet sich besonders zur Online-Durchführung. Nach Herstellung einer sicheren Datenverbindung zwischen Chipkarte 10 und Compilerserver 30 kann deshalb vorgesehen sein, daß der Nutzer unter einer Auswahl mehrerer möglicher Betriebsoptionen zur Weiterbearbeitung eine Auswahl treffen muß. In diesem Fall sendet der Compilerserver 30 nach Herstellung der sicheren Datenverbindung eine Anbietungsmitteilung über die möglichen Betriebsoptionen an den Nutzercomputer 20, Schritt 708. Aus den mitgeteilten Optionen wählt der Nutzer über den Nutzercomputer 20 die gewünschte aus, etwa eine Programmerstellung mit Online-Übersetzung, Schritt 710, oder einen Debug-Modus, in dem die Ausführbarkeit eines neu erzeugten Programmcodes online festgestellt wird.

Zur Erhöhung der Sicherheit der Datenübertragung zum Compilerserver 30 kann nachfolgend optional eine Signatur des Programmquelltextes Q durch die Chipkarte 10 vorgesehen sein, Schritt 711. Die Signatur erfolgt in an sich bekannter Weise, indem die Chipkarte 10 über den Quelltext Q einen Hash-

wert bildet und diesen mit dem privaten Schlüssel 130 der Chipkarte verschlüsselt. Die Hashwertbildung kann dabei, insbesondere bei nicht ausreichenden Hardwareresourcen auf einer Chipkarte 10, durch den Nutzercomputer 20 erfolgen.

Den, gegebenenfalls signierten Programmquelltextcode verschlüsselt die Chipkarte 10 mit dem zuvor bestimmten Sitzungsschlüssel SK_{SSL} zu einem Transportcode TQ_{SSL}, Schritt 712, den sie anschließend über den Nutzercomputer 20 an den Compilerserver 30 sendet, Schritt 714.

Jener entschlüsselt den eingegangenen Transportcode TQ_{SSL} wieder mit dem Sitzungsschlüssel SK_{SES}, Schritt 716, um den Programmquelltext Q wiederherzustellen. Falls eine Signatur vorhanden ist, prüft er durch erneute Bildung des Hashwertes unter Verwendung des öffentlichen Kartenschlüssels 332 deren Richtigkeit.

Aus dem wiederhergestellten Programmquelltext Q erzeugt der Compilerserver 30 anschließend durch Ausführung der Schritte 518, 520 einen ausführbaren Programmcode C.

Den erzeugten Programmcode C versieht der Compilerserver 30 mit einer Signatur, die er durch Bildung eines Hashwertes und Verschlüsseln des Hashwertes mit dem privaten Schlüssel 330 des Compilerservers 30 erzeugt. Den entstandenen, signierten Code verschlüsselt er sodann mit dem öffentlichen Schlüssel 332 der Chipkarte 10, Schritt 718. Das danach vorliegende Chiffrat überführt der Compilerserver 30 nachfolgend durch Verschlüsseln mit dem Sitzungsschlüssel SK_{SES} in ein Übergangsformat C_{SSL}, Schritt 720, das er als Transportcode schließlich an den Nutzercomputer 20 übermittelt, Schritt 722.

Dieser leitet den eingegangenen Transportcode C_{SSL} an die Chipkarte 10 weiter, Schritt 724 welche daraus durch Entschlüsselung mit dem Sitzungsschlüssel SK_{SES} wieder das Chiffrat des ausführbaren Programmcodes generiert, Schritt 725. Falls der Programmcode C im Compilerserver 30 signiert wurde, entschlüsselt die Chipkarte 10 das Chiffrat weiter mit dem privaten Schlüssel 130 der Chipkarte 10 und prüft die danach vorliegende Signatur mit dem öffentlichen Schlüssel 128 des Compilerservers 30, Schritt 726. Ist das Ergebnis der Signaturprüfung positiv, lädt die Chipkarte 10 den somit vorliegenden ausführbaren Programmcode C in die Speicheranordung 104 in den zur Aufnahme von Komplettierungsprogrammcode vorgesehenen Speicherraum 113, Schritt 728.

Wie bei der Ausführungsform nach Fig.5 wurde der in Fig. 7 dargestellte Verfahrensablauf der Klarheit wegen sequentiell beschrieben. Praktisch ist es jedoch sinnvoll, Verfahrensschritte, für die das möglich ist, quasiparallel auszuführen, indem Compilerserver 30 und Chipkarte 10 sie zeitlich überlagernd ausführen. Das gilt z.B. für die Schritte 712 bis 716, d.h. die chipkartenseitige Verschlüsselung und die compilerserverseitige Wiederherstellung des Programmquelltextes Q. Sie erfolgen zweckmäßig in Form eines kontinuierlichen, quasiparallelen Datenaustausch- und verarbeitungsprozesses, so daß der Programmquelltext Q nahezu unmittelbar nach Absendung des letzten Datensatzes durch die Chipkarte 10 im Compilerserver 30 vorliegt. Eine Zwischenspeicherung bis zum vollständigen Eingang des Programmquelltext Q erfolgt nicht. Weiter bietet sich eine Realisierung in Gestalt eines kontinuierlichen, quasiparallelen Datenaustausch- und verarbeitungsprozesses auch für die Schritte 718 bis 728 an, d.h. für die compilerserverseitige Verschlüsselung des ausführbaren Programmcodes C und seine chipkartenseitige Wiederherstellung sowie das Laden in den Speicherraum 113 auf der

Chipkarte 10. Die Ausführung dieser Schritte durch Compilerserver 30 und Chipkarte 10 erfolgt zweckmäßig ohne Zwischenspeicherung unmittelbar datensatzweise, so daß der ausführbare Programmcode C im wesentlichen unmittelbar nach Absendung des letzten Transportcodedatensatzes durch den Compilerserver 30 im Speicher der Chipkarte 10 vorliegt.

Im Rahmen einer Programmerstellung gemäß einer der vorstehend beschriebenen Ausführungsformen kann die Durchführung einer Debug-Routine vorgesehen sein. Damit wird ein durch den Compilerserver 30 erstellter Programmcode C vor dem Laden auf eine Chipkarte 10 auf Lauffähigkeit geprüft. Das Prinzip einer solchen Debug-Routine ist in Fig. 8 veranschaulicht, wobei zur Vereinfachung der Beschreibung die zur Sicherung der Datenübertragung gerichteten Maßnahmen, d.h. vor allem die verschiedenen Verschlüsselungen, nicht gezeigt sind.

Die Debug-Routine ist als Programm 316 im Compilerserver 30 angelegt und wird auch dort ausgeführt. Zusätzlich oder als Bestandteil des Programmes 316 beinhaltet sie eine einen Datenträger nachbildende Hardware zur Simulation und/oder eine softwaremäßige Nachbildung eines Datenträgers zur Emulation eines erzeugten Programmes auf dem Compilerserver 30 unter den auf dem Datenträger vorhandenen technischen Randbedingungen. Gesteuert wird sie, nach Einstellung eines entsprechenden Betriebsmodus im Compilerserver 30, über den Editor 22 im Nutzercomputer 20. Die Betriebsmoduseinstellung kann z.B. im Rahmen der Auswahl einer Betriebsoption in den Schritten 708 und 710 erfolgen, wenn die Programmerstellung gemäß der in Fig. 6 dargestellten Ausführungsform vorgenommen wird. Der Debug-Betriebsmodus gestattet es u.a., vom Nutzercomputer 20 aus ein im Compilerserver 30 erzeugtes Programm zu starten, Stopmarken zu setzen, Speicherbereiche anzuzeigen sowie Variablen auszulesen und zu setzen.

Für die Ausführung der Debug-Routine wird zunächst in üblicher Weise ein Programmquelltext Q erstellt, Schritt 504, und eine Verbindung zum Compilerserver 30 aufgebaut, Schritt 700. Anschließend wird der Quelltext Q gemäß einer der zuvor beschriebenen Ausführungsformen an den Compilerserver 30 übermittelt, Schritt 800.

Ist der Quelltext Q eingegangen, bietet der Compilerserver 30 dem Nutzer die Erzeugung eines Programmcodes C im Debug-Betriebsmodus an, Schritt 802. Ein Nutzer kann den Modus darauf über den Nutzercomputer 20 auswählen, Schritt 804. Wurde der Debug-Betriebsmodus gewählt, erstellt der Compilerserver 30 aus dem eingegangenen Programmquelltext Q durch Ausführung der Schritte 526, 528 einen vorläufigen Programmcode C_V, der auf der im Compilerserver 30 vorhandenen Simulations- und/oder Emulationsumgebung lauffähig ist. Den vorläufigen Programmcode C_V speichert der Compilerserver 30 in einen Zwischenspeicher, Schritt 806. Anschließend übermittelt er dem Nutzercomputer 20 eine Erstellungsmeldung, Schritt 808, die dieser zur Anzeige bringt, Schritt 810.

Der Nutzer kann das Quelltextprogramm Q nun mittels des Nutzercomputers 20 mit Debug-Anweisungen versehen, d.h. Stopmarken setzen, die Ausführung eines Programmes in Einzelschritten oder das Anzeigen von Variablen veranlassen, Schritt 812. Die Debug-Anweisungen werden dem Compilerserver 30 mitgeteilt.

Nachfolgend kann über den Nutzercomputer 20 die Ausführung des durch den vorläufigen Programmcode C_V realisierten Programmes auf dem Compilerserver 30 ausgelöst werden, Schritt 814. Der Compilerserver 30 führt darauf das Programm unter Berücksichtigung der zuvor mitgeteilten Debug-

Anweisungen aus, Schritt 816. Jeweils nach Ausführung eines durch die Debug-Anweisungen festgelegten Programmabschnittes übermittelt er eine Ergebnismeldung an den Nutzercomputer 20, Schritt 818, die dieser zur Anzeige bringt, Schritt 820. Abhängig von den übergebenen Debug-Anweisungen kann darauf ein Eingriff eines Nutzers in die Programmausführung vorgesehen sein, etwa durch Eingabe von Variablen oder durch Setzen neuer Debug-Anweisungen, Schritt 822. Gegebenenfalls vorgenommene Eingriffe in den Programmquelltext Q oder neue Debug-Anweisungen, übermittelt der Nutzercomputer 20 dem Compilerserver 30. Ist eine Debug-Anweisung schließlich abgearbeitet, übermittelt der Nutzercomputer 20 dem Compilerserver 30 ein Fortsetzungssignal, Schritt 824, auf das hin jener durch Wiederholung des Schrittes 814 die Ausführung des nächsten Programmabschnittes veranlaßt. Dabei berücksichtigt er eventuell vorgenommene Eingriffe in den Programmquelltext Q oder neue Debug-Anweisungen. Die Schritte 814 bis 824 werden wiederholt, bis der Compilerserver 30 ein durch einen vorläufigen Programmcode C_V realisiertes Programm vollständig ausgeführt hat.

Erweist sich das Programm schließlich als fehlerfrei lauffähig, veranlaßt der Nutzer über den Nutzercomputer 20 einen Wechsel des Betriebsmodus in den Standard-Modus, Schritt 826. Der Compilerserver 30 erzeugt daraufhin aus dem zu diesem Zeitpunkt vorliegenden Programmquelltext Q einen lauffähigen Programmcode C und überträgt diesen wie anhand der Fig. 4 bis 6 beschrieben über den Nutzercomputer 20 an die Chipkarte 10, Schritt 828. Desweiteren löscht er den zwischengespeicherten, vorläufigen Programmcode C_V, Schritt 830

Die vorstehend beschriebene Folge von Ausführungsbeispielen ist jeweils als Basis für eine konkrete Verfahrensrealisierung zu verstehen. Unter Beibehal-

tung des grundlegenden Ansatzes, zur Erzielung einer sicheren Programmerstellung vorkomplettierte Datenträger zu verwenden, sind die Ausführungsbeispiele jeweils in einem weiten Rahmen ausgestaltbar. Dies gilt insbesondere für die Ausführung der Strukturelemente, d.h. der Chipkarte , des Nutzercomputers, des Datennetzes und des Compilerservers. Weiter können die genannten Verschlüsselungs- und Authentifizierungsverfahren selbstverständlich durch andere mit gleicher Sicherheitswirkung ersetzt werden. Alle beschriebenen Ausführungsformen lassen sich insbesondere durch die Verwendung weiterer Schlüssel, Sequenzzähler oder anderer kryptografischer Algorithmen auf eine nochmals erhöhte Sicherheitsstufe bringen. Aus technischen oder aus Sicherheitsgründen können auch weitere Umformatierungen vorgesehen sein. So ist es insbesondere bei programmierbaren Chipkarten mit Blick auf deren begrenzten Speicherplatz üblich, einen auf einem Compilerserver 30 erzeugten lauffähigen Programmcode vor oder beim Laden in die Chipkarte nochmals speicheroptimierend umzuformatieren, indem beispielsweise symbolische Referenzierungen durch absolute Adressen ersetzt werden. Aus Gründen der Übersichtlichkeit wurden ferner jeweils nur Gutfälle beschrieben. Die Behandlung von Fehlerfällen läßt sich daraus jedoch unter Verwendung bekannter Standardlösungen ableiten.

Patentansprüche

1. Verfahren zur verteilten Erstellung eines ausführbaren Programmes für einen programmierbaren, tragbaren Datenträger, wobei die Erstellung eines Programmquelltextes auf einem ersten, bei einem Nutzer befindlichen Computer, Compilieren und Linken des Programmquelltextes zu einem ausführbaren Programmcode nach Übertragung auf einem zweiten, beim Herausgeber des Datenträgers befindlichen Computer, und das Laden des ausführbaren Programmcodes in den Datenträger nach Rückübertragung wieder über den ersten Computer erfolgt, dadurch **gekennzeichnet**, daß

   in einem Vorkomplettierungsschritt auf dem Datenträger (10) Softwarewerkzeuge zur Endbearbeitung angelegt werden, die es erlauben, aus einem in einem Übergangsformat vorliegenden Transportcode (U, UC_{SM}, C_{SSL},) einen ausführbaren Programmcode (C) zu gewinnen,

   im zweiten Computer (30) erzeugter, ausführbarer Programmcode (C) für die Rückübertragung in Transportcode (U, UC_{SM}, C_{SSL},) umgewandelt wird, und

   an den ersten Computer (20) zur Einbringung in den Datenträger (10) rückübertragener Transportscode (U, UC_{SM}, C_{SSL},) mittels der Softwarewerkzeuge in ausführbaren Programmcode (C) rückgewandelt wird.

2. Verfahren nach Anspruch 1, dadurch **gekennzeichnet**, daß der auf dem ersten Computer (20) erstellte Programmquelltext (Q) für die Übertragung zum zweiten Computer (30) eine Transportsicherung erhält, indem er verschlüsselt wird.

3. Verfahren nach Anspruch 1, dadurch **gekennzeichnet**, daß die Softwarewerkzeuge eine einen Datenträger (10) bezeichnende Identifikationinformation (114) sowie einen Signaturschlüssel (124) beinhalten.

4. Verfahren nach Anspruch 1, dadurch **gekennzeichnet**, daß die Softwarewerkzeuge ein Programm zur Durchführung eines SSL-Handshake-Protokolls (122) und /oder ein Programm zur Durchführung von Secure-Messaging (120) beinhalten.

5. Verfahren nach Anspruch 1, dadurch **gekennzeichnet**, daß die Softwarewerkzeuge einen privaten Datenträgerschlüssel (130) sowie ein Programm zur Prüfung einer Signatur mit dem öffentlichen Schlüssel (128) eines zweiten Computers (30) beinhalten.

6. Verfahren nach Anspruch 1, dadurch **gekennzeichnet**, daß in den tragbaren Datenträger (10) zu ladender, ausführbarer Programmcode (C) unter Einbeziehung des zweiten Computers (30) ausgeführt wird, um mögliche Fehler zu ermitteln.

7. Verfahren nach Anspruch 6, dadurch **gekennzeichnet**, daß in den tragbaren Datenträger (10) zu ladender, ausführbarer Programmcode (C) auf dem zweiten Computer (30) unter Einbeziehung des ersten Computers (20) ausgeführt wird.

8. Verfahren nach Anspruch 1, dadurch **gekennzeichnet**, daß der ausführbare Programmcode (C) im Speicher (113) des Datenträgers (10) abgelegt wird.

9. Programmierbarer tragbarer Datenträger mit einem integrierten Schaltkreis, welcher einen Prozessor sowie einen Speicher zur Aufnahme von durch den Prozessor ausführbarem Programmcode aufweist, dadurch **gekennzeichnet**, daß in dem integrierten Schaltkreis (12) Softwarewerkzeuge zur Endbearbeitung angelegt sind, die es ermöglichen, einen in einem Übergangsformat zugeführten Transportcode (U, UC_{SM}, C_{SSL}) in ausführbaren Programmcode (C) zu überführen.

10. Datenträger nach Anspruch 9, dadurch **gekennzeichnet**, daß er wenigstens einen Sequenzzähler (136) aufweist.

11. Datenträger nach Anspruch 9, dadurch **gekennzeichnet**, daß er eine den Datenträger (10) bezeichnende Identifikationsinformation (114) sowie einen Schlüssel (124) zur Bildung eines Datensicherungscodes enthält, welche ihn einem definierten zweiten Computer (30) zuordnen.

12. Computer zur Durchführung einer verteilten Erstellung eines ausführbaren Programmes für einen programmierbaren, tragbaren Datenträger, enthaltend zumindest ein Compilierungsprogramm sowie ein Linkprogramm (312), dadurch **gekennzeichnet**, daß er über Mittel verfügt, um aus einen zugegangenen, in einem Übergangsformat vorliegenden Transportcode (T, TQ, TQ_{SSL},) einen Programmquelltext (Q) rückzugewinnen.

13. Computer nach Anspruch 12, dadurch **gekennzeichnet**, daß er Mittel aufweist, um die Identität eines zu programmierenden Datenträgers (10) festzustellen und zu überprüfen.

14. Computer nach Anspruch 12, dadurch **gekennzeichnet**, daß er eine Tabelle (340) führt, in der für jeden tragbaren Datenträger (10), der mittels des Computers (30) programmierbar ist, eine den Datenträger bezeichnende Identifikationsinformation (114) abgelegt ist.

15. Computer nach Anspruch 12, dadurch **gekennzeichnet**, daß er über Mittel (321) zur formalen Verifikation eines durch Compilierung erzeugten Programmcodes (C) aufweist.

16. Computer nach Anspruch 12, dadurch **gekennzeichnet**, daß er über Mittel (316) verfügt, um einen erzeugten Programmcode (C) durch unmittelbare Ausführung darauf zu prüfen, ob er Fehler enthält.

17. Computer zur Durchführung einer verteilten Erstellung eines ausführbaren Programmes für einen programmierbaren, tragbaren Datenträger, enthaltend zumindest eine erste Schnittstelle für einen Datenaustausch mit einem Datenträger sowie eine zweite Schnittstelle zu einer Datenverbindung, dadurch **gekennzeichnet**, daß

er Mittel aufweist, um ein Editierungsprogramm (22) zur Erstellung eines durch den Computer (20) selbst nicht ausführbaren Programmquelltextes (Q) auszuführen,

und daß er Mittel aufweist, um über die erste und die zweite Schnittstelle (24, 26) eine direkte Datenübertragung zwischen einem tragbaren Datenträger (10) und einem über die Datenverbindung (28) angeschlossenen, zweiten Computer (39) zu ermöglichen.

18. Computer nach Anspruch 17, dadurch **gekennzeichnet**, daß er Mittel aufweist, um ein SSL-Protokoll auszuführen.

19. System zur verteilten Erstellung eines ausführbaren Programmes für einen programmierbaren, tragbaren Datenträger, beinhaltend einen tragbaren Datenträger gemäß Anspruch 9 sowie einem Computer gemäß Anspruch 12.

20. System nach Anspruch 20, dadurch **gekennzeichnet**, daß es weiterhin einen Computer gemäß Anspruch 17 umfaßt.

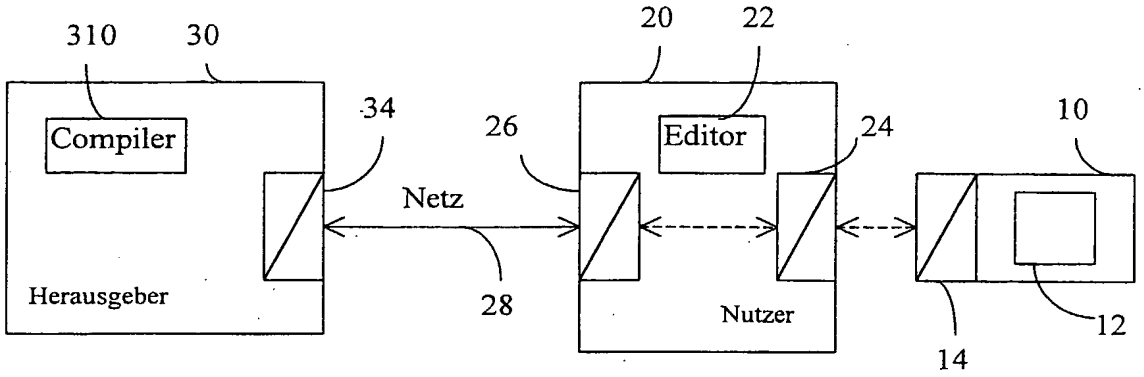

310
30
20
22
34
26
Editor
24
10
Compiler
Netz
Herausgeber
28
Nutzer
12
14

Fig.1

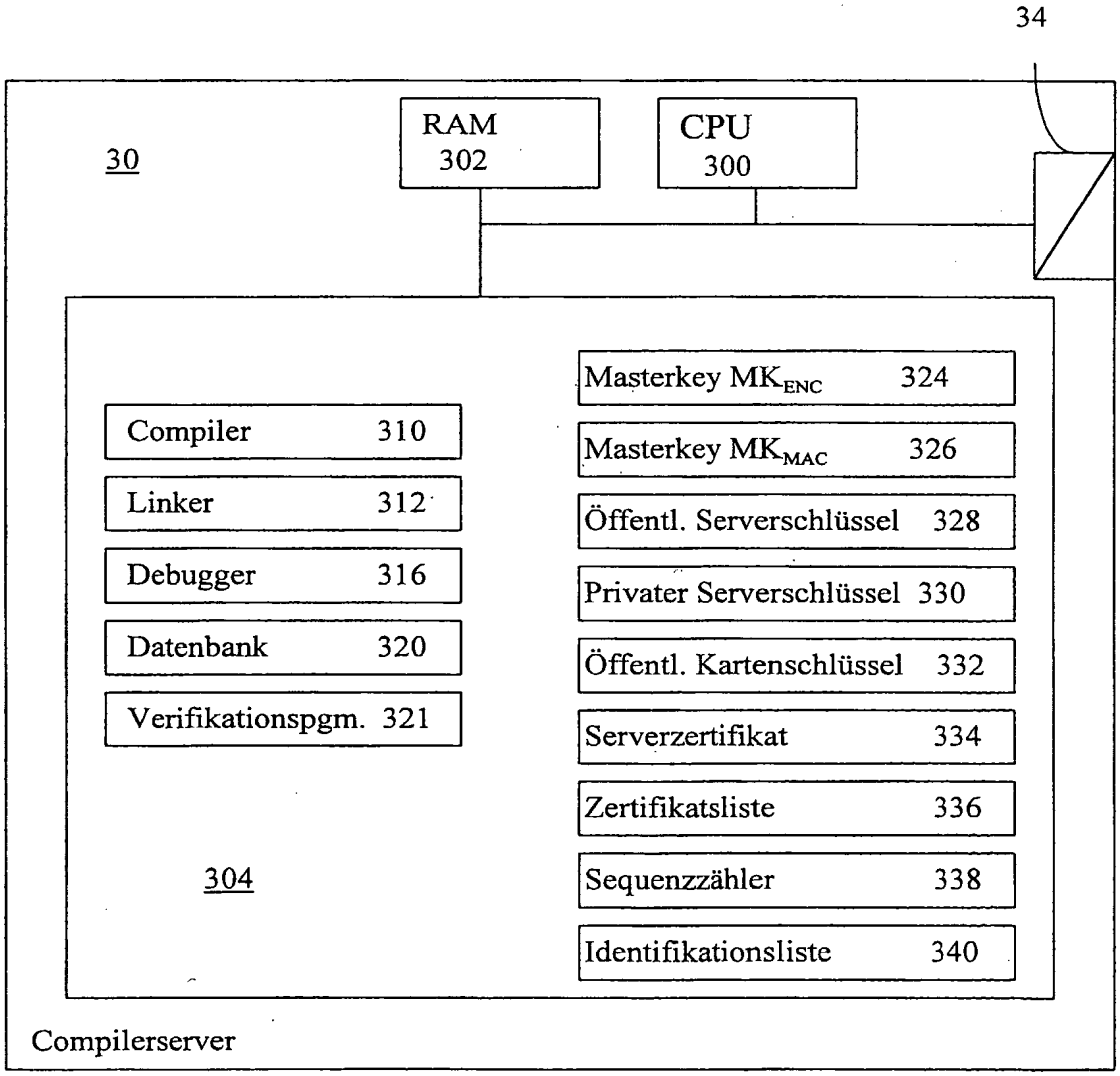
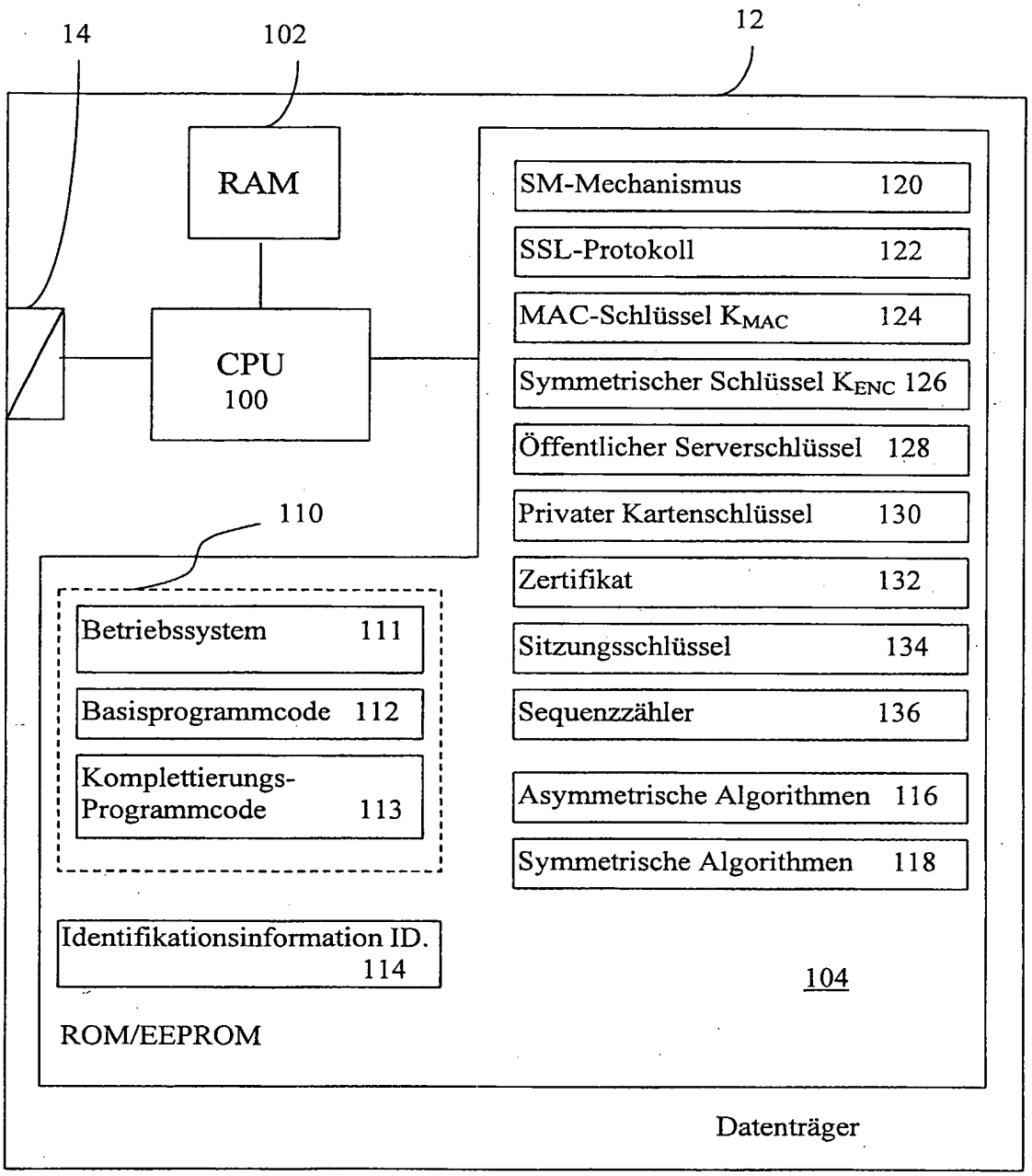

34
30
RAM
302
CPU
300
Compiler 310
Linker 312
Debugger 316
Datenbank 320
Verifikationspgm. 321
304
Masterkey MK_{ENC} 324
Masterkey MK_{MAC} 326
Öffentl. Serverschlüssel 328
Privater Serverschlüssel 330
Öffentl. Kartenschlüssel 332
Serverzertifikat 334
Zertifikatsliste 336
Sequenzzähler 338
Identifikationsliste 340
Compilerserver

Fig. 3

Fig.2


14
102
12
RAM
CPU
100
SM-Mechanismus 120
SSL-Protokoll 122
MAC-Schlüssel K_{MAC} 124
Symmetrischer Schlüssel K_{ENC} 126
Öffentlicher Serverschlüssel 128
Privater Kartenschlüssel 130
Zertifikat 132
Sitzungsschlüssel 134
Sequenzzähler 136
110
Betriebssystem 111
Basisprogrammcode 112
Komplettierungs-Programmcode 113
Asymmetrische Algorithmen 116
Symmetrische Algorithmen 118
Identifikationsinformation ID. 114
104
ROM/EEPROM
Datenträger

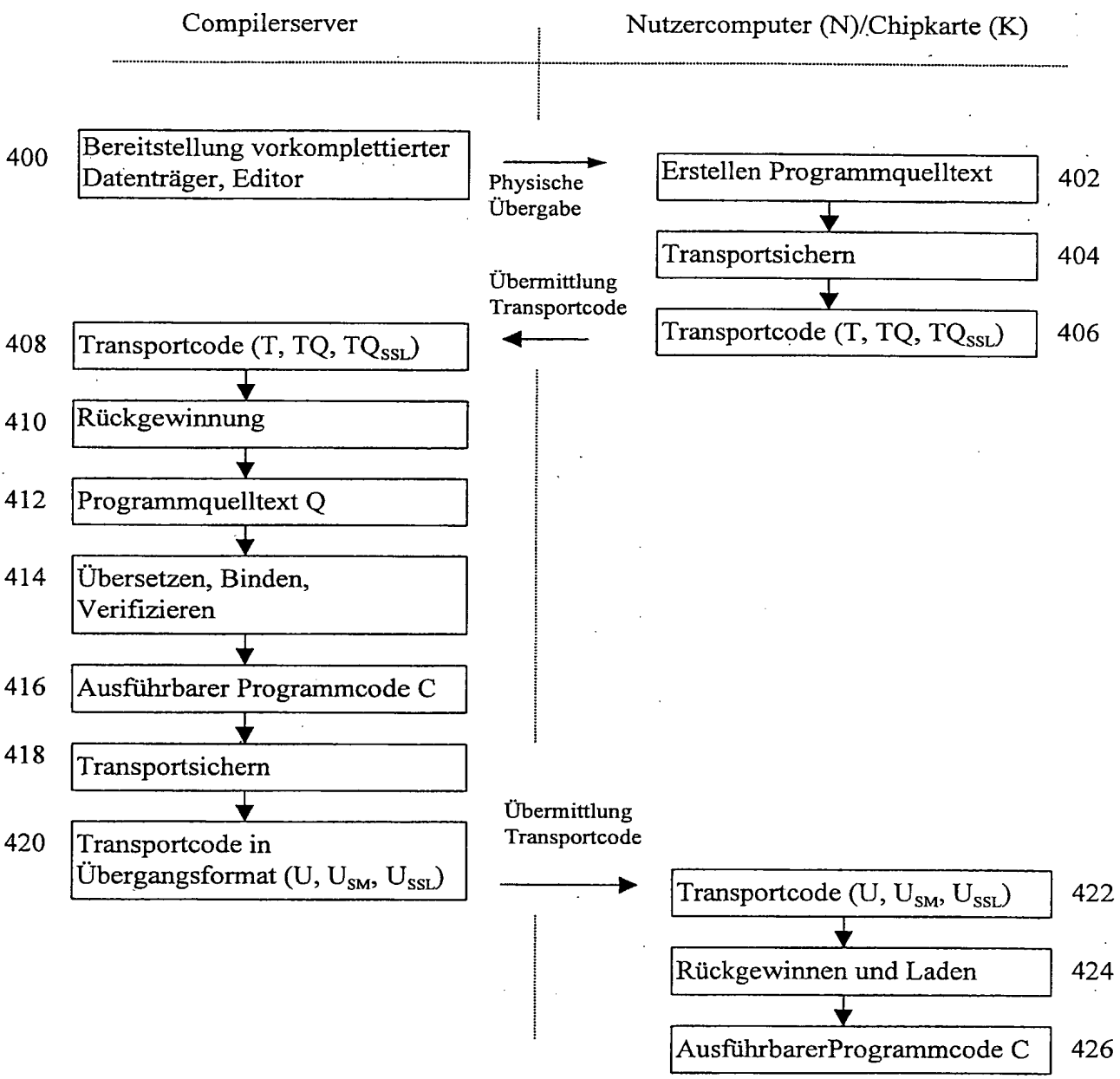

Compilerserver
Nutzercomputer (N)/Chipkarte (K)
400
Bereitstellung vorkomplettierter Datenträger, Editor
Physische Übergabe
Erstellen Programmquelltext
402
Transportsichern
404
Übermittlung Transportcode
Transportcode (T, TQ, TQSSL)
406
408
Transportcode (T, TQ, TQSSL)
410
Rückgewinnung
412
Programmquelltext Q
414
Übersetzen, Binden, Verifizieren
416
Ausführbarer Programmcode C
418
Transportsichern
420
Transportcode in Übergangsformat (U, USM, USSL)
Übermittlung Transportcode
Transportcode (U, USM, USSL)
422
Rückgewinnen und Laden
424
AusführbarerProgrammcode C
426

Fig. 4

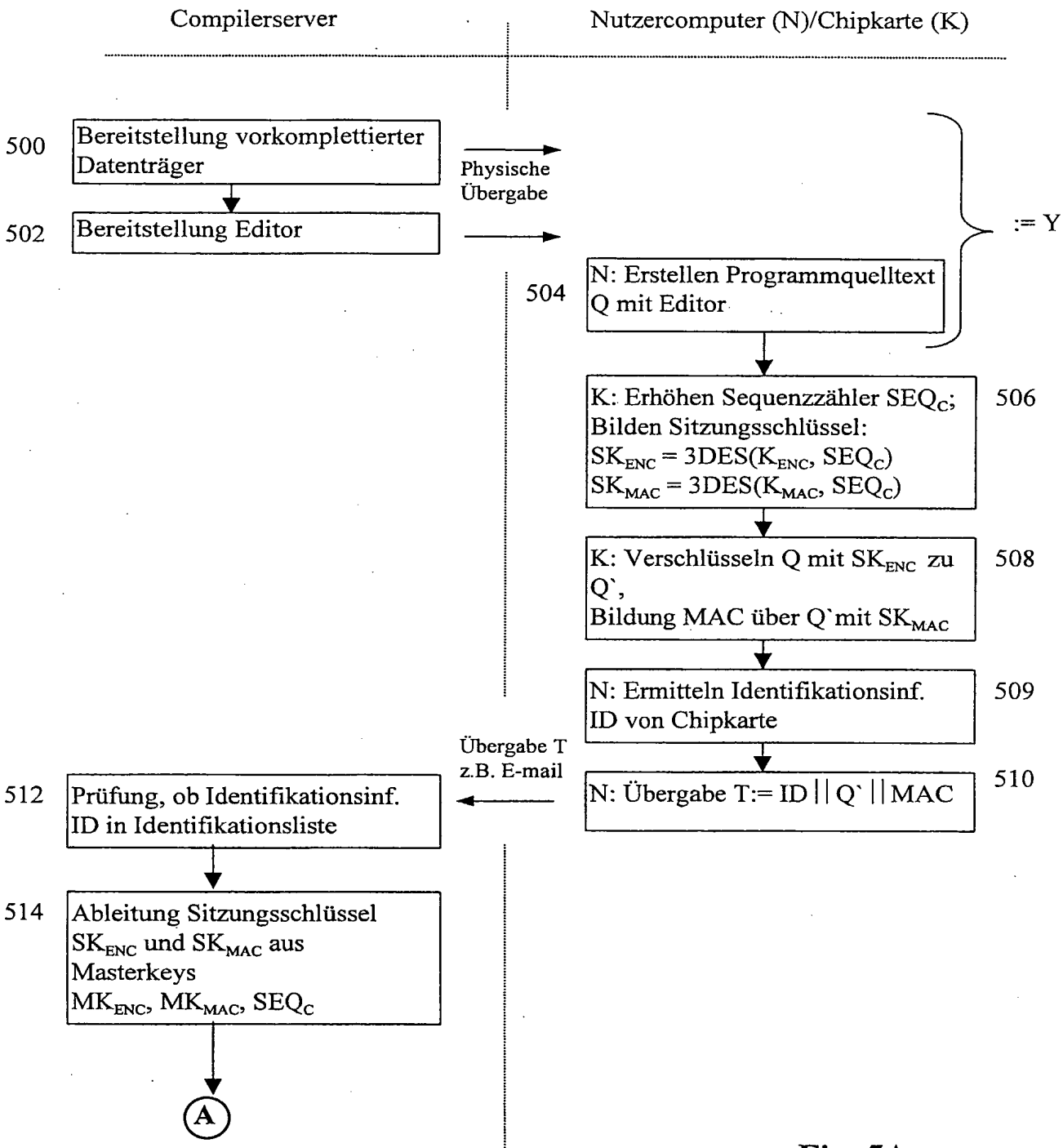

Compilerserver
Nutzercomputer (N)/Chipkarte (K)
500
Bereitstellung vorkomplettierter Datenträger
Physische Übergabe
502
Bereitstellung Editor
:= Y
504
N: Erstellen Programmquelltext Q mit Editor
K: Erhöhen Sequenzzähler SEQC; Bilden Sitzungsschlüssel:
SKENC = 3DES(KENC, SEQC)
SKMAC = 3DES(KMAC, SEQC)
506
K: Verschlüsseln Q mit SKENC zu Q`,
Bildung MAC über Q`mit SKMAC
508
N: Ermitteln Identifikationsinf. ID von Chipkarte
509
Übergabe T z.B. E-mail
N: Übergabe T:= ID || Q` || MAC
510
512
Prüfung, ob Identifikationsinf. ID in Identifikationsliste
514
Ableitung Sitzungsschlüssel SKENC und SKMAC aus Masterkeys
MKENC, MKMAC, SEQC
A

Fig. 5A

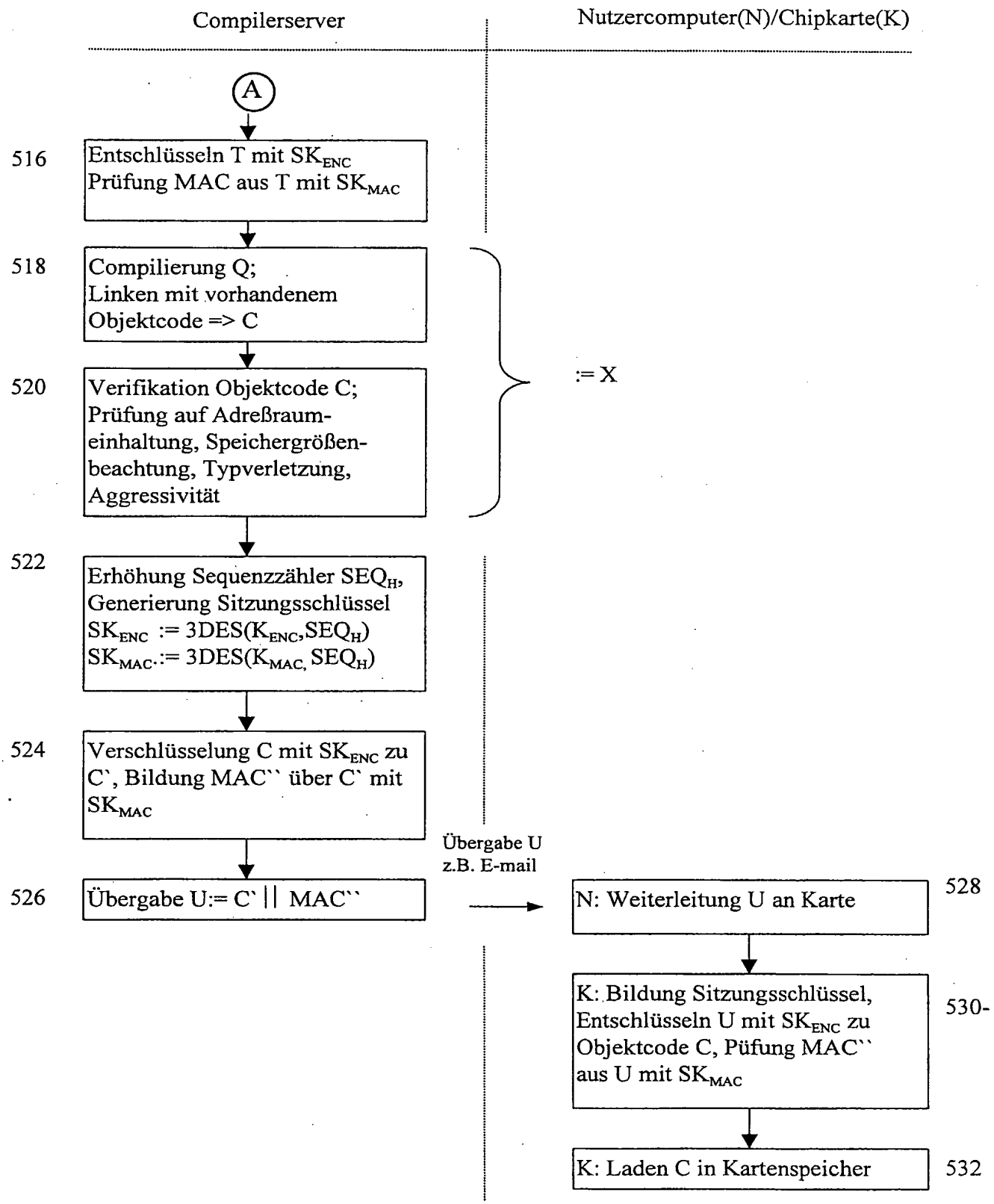

Compilerserver
Nutzercomputer(N)/Chipkarte(K)
A
516
Entschlüsseln T mit SK_{ENC}
Prüfung MAC aus T mit SK_{MAC}
518
Compilierung Q;
Linken mit vorhandenem Objektcode => C
520
Verifikation Objektcode C;
Prüfung auf Adreßraum-einhaltung, Speichergrößen-beachtung, Typverletzung, Aggressivität
:= X
522
Erhöhung Sequenzzähler SEQ_H,
Generierung Sitzungsschlüssel
SK_{ENC} := 3DES(K_{ENC},SEQ_H)
SK_{MAC}.:= 3DES(K_{MAC}, SEQ_H)
524
Verschlüsselung C mit SK_{ENC} zu C`, Bildung MAC`` über C` mit SK_{MAC}
Übergabe U
z.B. E-mail
526
Übergabe U:= C` || MAC``
N: Weiterleitung U an Karte
528
K: Bildung Sitzungsschlüssel, Entschlüsseln U mit SK_{ENC} zu Objektcode C, Püfung MAC`` aus U mit SK_{MAC}
530
K: Laden C in Kartenspeicher
532

Fig. 5B

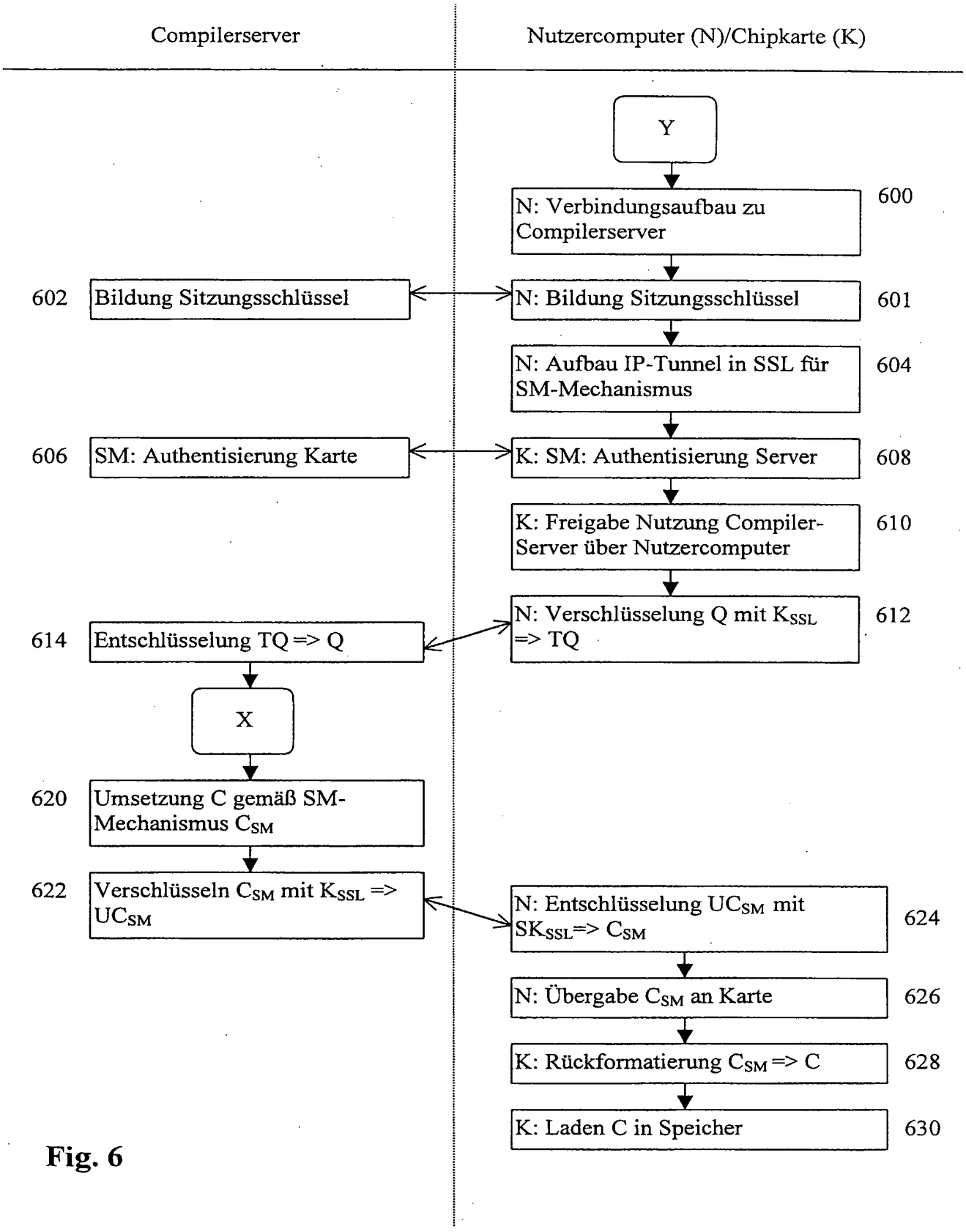

Compilerserver
Nutzercomputer (N)/Chipkarte (K)
Y
N: Verbindungsaufbau zu Compilerserver
600
602
Bildung Sitzungsschlüssel
N: Bildung Sitzungsschlüssel
601
N: Aufbau IP-Tunnel in SSL für SM-Mechanismus
604
606
SM: Authentisierung Karte
K: SM: Authentisierung Server
608
K: Freigabe Nutzung Compiler-Server über Nutzercomputer
610
N: Verschlüsselung Q mit K_{SSL} => TQ
612
614
Entschlüsselung TQ => Q
X
620
Umsetzung C gemäß SM-Mechanismus C_{SM}
622
Verschlüsseln C_{SM} mit K_{SSL} => UC_{SM}
N: Entschlüsselung UC_{SM} mit SK_{SSL}=> C_{SM}
624
N: Übergabe C_{SM} an Karte
626
K: Rückformatierung C_{SM} => C
628
K: Laden C in Speicher
630

Fig. 6

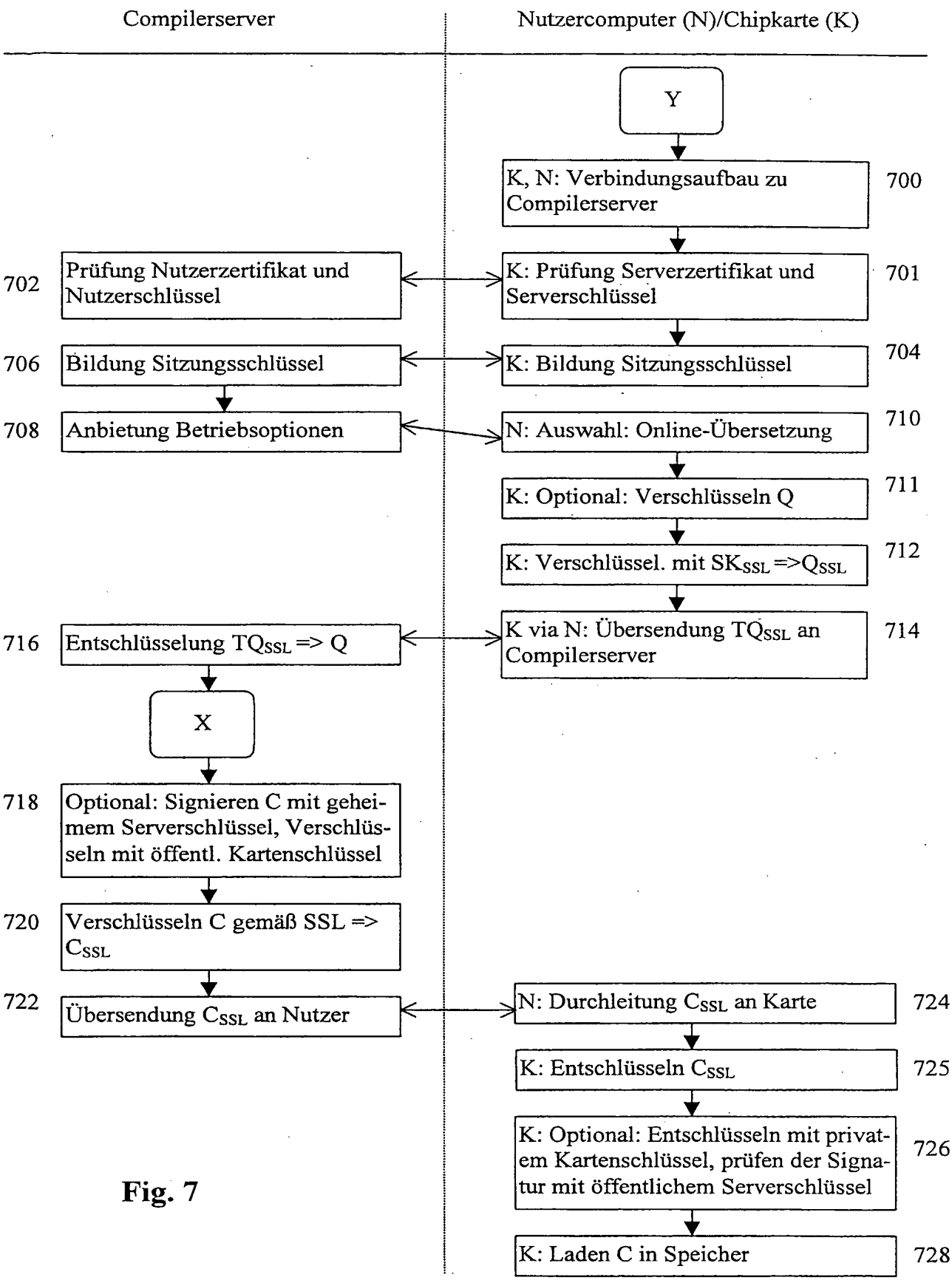

Compilerserver
Nutzercomputer (N)/Chipkarte (K)
Y
K, N: Verbindungsaufbau zu Compilerserver
700
702
Prüfung Nutzerzertifikat und Nutzerschlüssel
K: Prüfung Serverzertifikat und Serverschlüssel
701
706
Bildung Sitzungsschlüssel
K: Bildung Sitzungsschlüssel
704
708
Anbietung Betriebsoptionen
N: Auswahl: Online-Übersetzung
710
K: Optional: Verschlüsseln Q
711
K: Verschlüssel. mit SK_{SSL} => Q_{SSL}
712
716
Entschlüsselung TQ_{SSL} => Q
K via N: Übersendung TQ_{SSL} an Compilerserver
714
X
718
Optional: Signieren C mit geheimem Serverschlüssel, Verschlüsseln mit öffentl. Kartenschlüssel
720
Verschlüsseln C gemäß SSL => C_{SSL}
722
Übersendung C_{SSL} an Nutzer
N: Durchleitung C_{SSL} an Karte
724
K: Entschlüsseln C_{SSL}
725
K: Optional: Entschlüsseln mit privatem Kartenschlüssel, prüfen der Signatur mit öffentlichem Serverschlüssel
726
K: Laden C in Speicher
728

Fig. 7

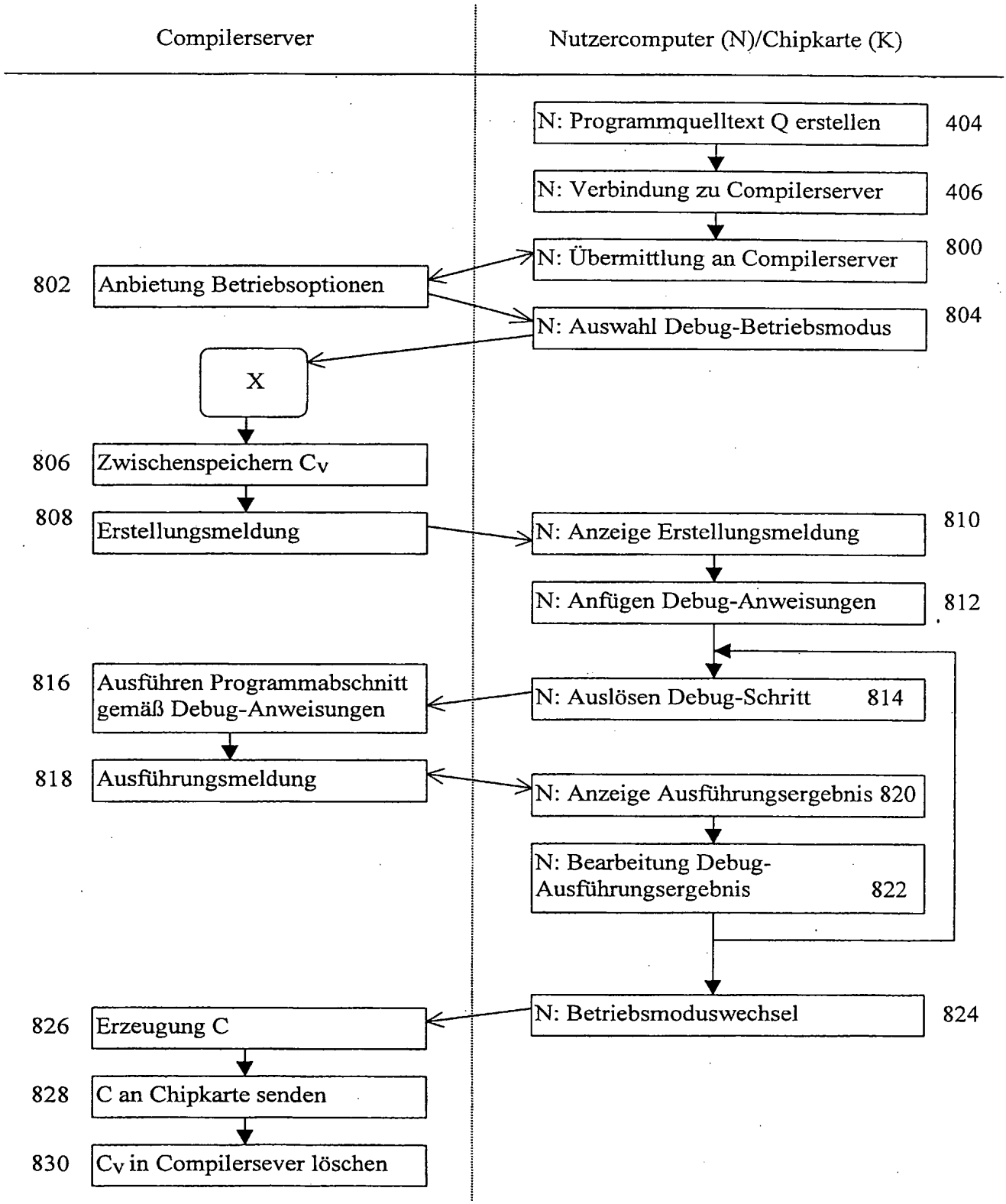

Compilerserver
Nutzercomputer (N)/Chipkarte (K)
N: Programmquelltext Q erstellen
404
N: Verbindung zu Compilerserver
406
N: Übermittlung an Compilerserver
800
802
Anbietung Betriebsoptionen
N: Auswahl Debug-Betriebsmodus
804
X
806
Zwischenspeichern Cv
808
Erstellungsmeldung
N: Anzeige Erstellungsmeldung
810
N: Anfügen Debug-Anweisungen
812
816
Ausführen Programmabschnitt gemäß Debug-Anweisungen
N: Auslösen Debug-Schritt
814
818
Ausführungsmeldung
N: Anzeige Ausführungsergebnis
820
N: Bearbeitung Debug-Ausführungsergebnis
822
N: Betriebsmoduswechsel
824
826
Erzeugung C
828
C an Chipkarte senden
830
Cv in Compilersever löschen

Fig. 8